Государственное профессиональное образовательное
автономное учреждение Амурской области
«БЛАГОВЕЩЕНСКИЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»
(**ГПОАУ СПО «БПК»**)

Е.П. Мунгалова

ЛАБОРАТОРНЫЕ РАБОТЫ

ПМ 02 Разработка, внедрение и адаптация программного обеспечения отраслевой направленности

Тема: Программирование на встроенных алгоритмических языках

программы подготовки специалистов среднего звена (ППССЗ)

специальности 09.02.05 Прикладная информатика (по отраслям)

Практикум

Благовещенск, 2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В практикуме приведены сгруппированные по занятиям упражнения по программированию на языке Java, а также методические указания, контрольные вопросы и задачи, предназначенные для решения студентами, изучающими курс «Программирование на встроенных алгоритмических языках».

Для каждого практического занятия определена область изучаемых Java- технологий и цели практического занятия.

Лабораторная работа № 1

«Создание и ведение проекта. Простейшая программа. Класс Math, его методы и константы»

Цель: освоить основные способы создания Java-программ, научиться создавать программы на Java, освоить основные понятия объектно-ориентированного программирования на Java.

Задачи:

1. Освоить основные способы создания Java-программ либо с помощью обычного редактора, либо с помощью среды разработки.
2. Используя примеры программ, приведенные в работах, познакомиться с основными приемами в программировании на Java.
3. Приобрести навыки в использовании системы помощи для поиска нужной информации по различным классам Java.
4. Научиться создавать программы на Java с использованием примером кодирования из приведенных примеров.
5. Освоить основные понятия объектно-ориентированного программирования на Java.

Общие компетенции:

ОК 1. Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.

ОК 4. Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.

ОК 5. Использовать информационно-коммуникационные технологии в профессиональной деятельности.

ОК 8. Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации.

Профессиональные компетенции:

ПК 2.2. Разрабатывать и публиковать программное обеспечение и информационные ресурсы отраслевой направленности со статическим и динамическим контентом на основе готовых спецификаций и стандартов.

ПК 2.3. Проводить отладку и тестирование программного обеспечения отраслевой направленности.

ПК 2.4. Проводить адаптацию отраслевого программного обеспечения.

ПК 2.5. Разрабатывать и вести проектную и техническую документацию.

ПК 2.6. Участвовать в измерении и контроле качества продуктов.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Запуск NetBeans и вывод текста на консоль

Запустим NetBeans:

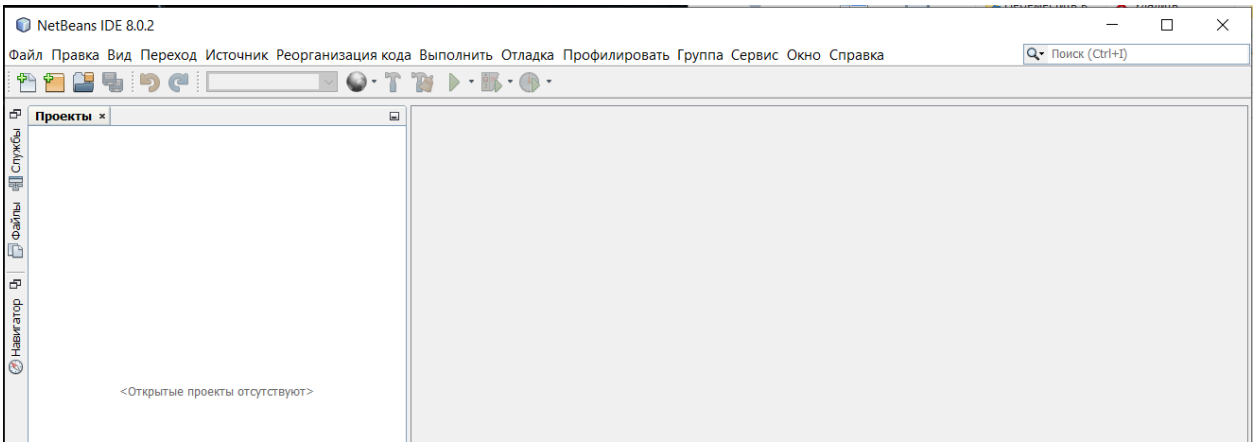


Рисунок 1 - Среда NetBeans

Среда NetBeans довольно интеллектуально понятна, и с ней очень легко работать. Создадим новый проект. Для этого выберем в меню пункт **Файл – Создать проект....** После этого перед нами откроется окно создания нового проекта:

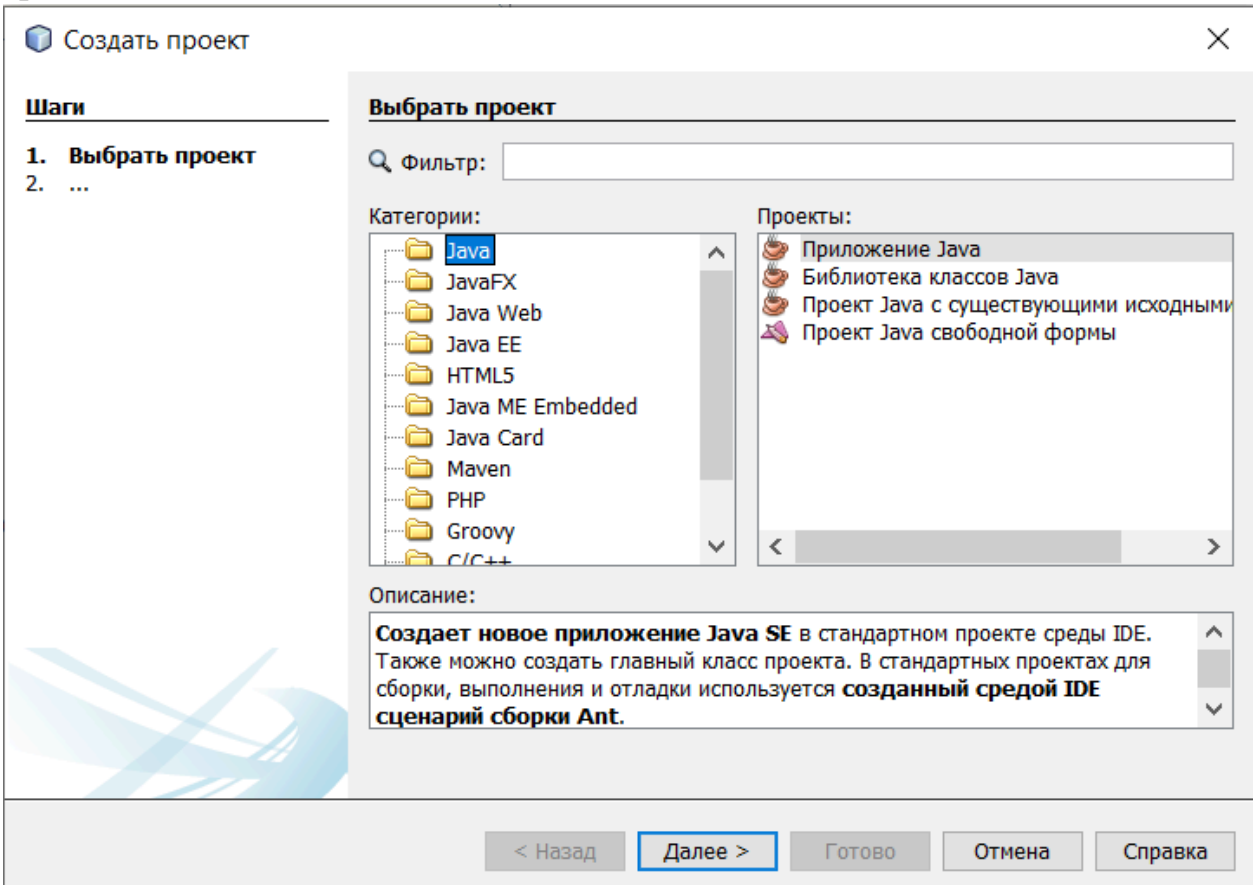


Рисунок 2 - Окно создания нового проекта

В окне создания нового проекта установите выбор как на рисунке 2 и нажмите кнопку **Далее >**.

Затем откроется окно настроек проекта (рис. 3), где в поле «**Имя проекта**» дадим проекту какое-нибудь название (например, *MyFirstProgramm*). Для всех остальных полей можно оставить значения по умолчанию. Последнее поле "**Создать главный класс**" указывает, что

автоматически в проекте будет создан класс программы *MyFirstProgramm*, который будет находиться в одноименном пакете *myfirstprogramm*.

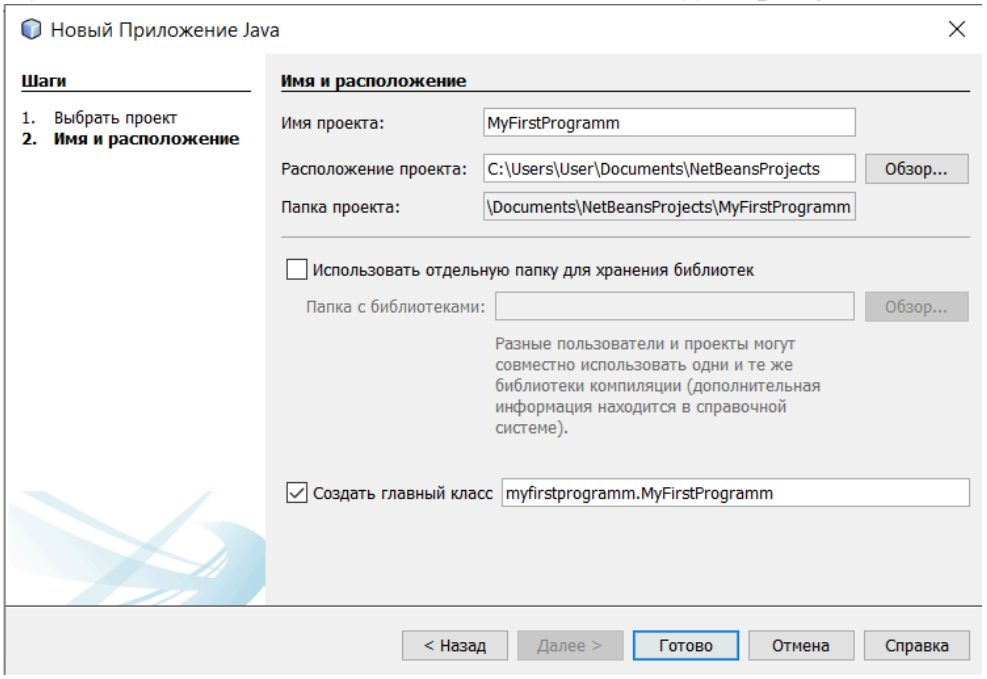


Рисунок 3 - Имя проекта

И в завершении создания проекта нажмем на кнопку **Готово**. И перед нами откроется новый проект в Netbeans:

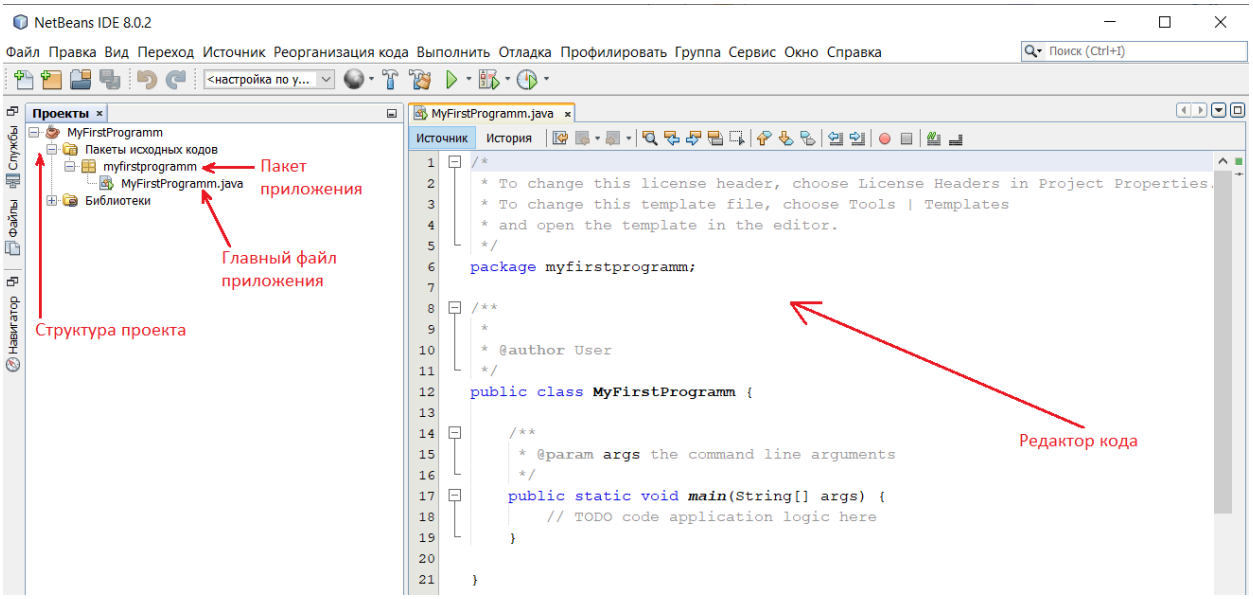


Рисунок 4 - Новый проект MyFirstProgramm

Узкое окно слева отображает все открытые проекты и их структуру. Окно справа отображает весь программный код и комментарии.

Изменим код по умолчанию на следующий, чтобы программа выводила простейшую строку на консоль:

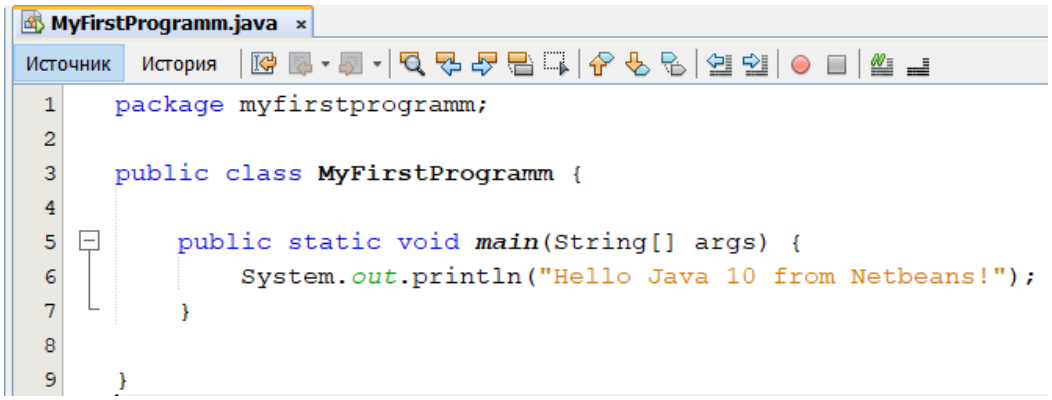


Рисунок 5 - Первая программа "Hello Java 10 from Netbeans!"

Теперь запустим проект на выполнение. Для этого на панели инструментов NetBeans нажмем на зеленую стрелочку (либо можно нажать на клавишу F6, либо выбрать в меню пункт **Выполнить** – **Запустить проект**). И внизу NetBeans откроется окно вывода, в котором можно наблюдать результаты программы:

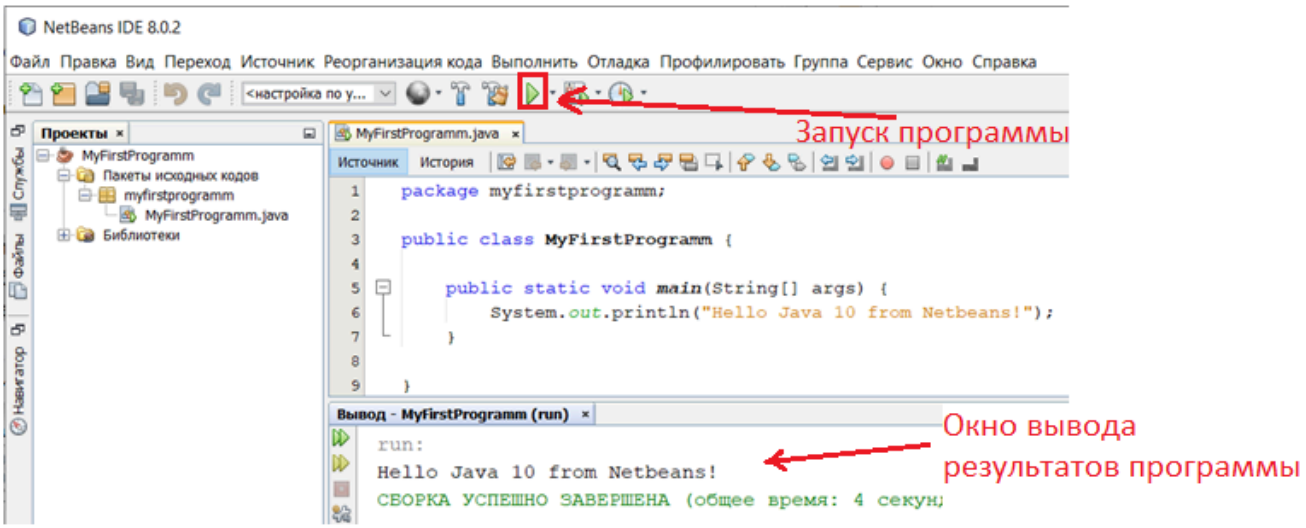


Рисунок 6 - Запуск и исполнение первой программы

Данная строка представляет вызов метода System.out.println, который выводит на консоль строку "Hello Java 10 from Netbeans!".

Для хранения данных в программе предназначены **переменные**. Переменная представляет именованную область памяти, которая хранит значение определенного типа. Каждая переменная имеет тип, имя и значение. Тип определяет, какую информацию может хранить переменная или диапазон допустимых значений.

Переменные объявляются следующим образом:

```
тип_данных имя_переменной;
```

Например, определим переменную, которая будет называться **X** и будет иметь тип **int**:

```
int x;
```

В этом выражении мы объявляем переменную **X** типа **int**. То есть **X** будет хранить некоторое число не больше 4 байт.

В качестве имени переменной может выступать любое произвольное название, которое удовлетворяет следующим требованиям:

- имя может содержать любые алфавитно-цифровые символы, а также знак подчеркивания, при этом первый символ в имени не должен быть цифрой
- в имени не должно быть знаков пунктуации и пробелов
- имя не может быть ключевым словом языка Java.

Объявив переменную, мы можем присвоить ей значений:

```
int x;        // объявление переменной
x = 10;       // присвоения значения
System.out.println(x); // вывод значения x, т.е. число 10
```

Также можно присвоить значение переменной при ее объявлении. Этот процесс называется **инициализацией**:

```
int x = 10; // объявление и инициализация переменной
System.out.println(x);  // 10
```

Ввод данных с консоли

Для получения ввода с консоли в классе **System** определен объект **in**. Однако непосредственно через объект `System.in` не очень удобно работать, поэтому, как правило, используют класс **`Scanner`**, который, в свою очередь использует `System.in`.

Так как класс Scanner находится в пакете java.util, то вначале его следует импортировать с помощью инструкции import java.util.Scanner или создать конструкцию Scanner, после чего слева появится желтая лампочка, нажав на которую появятся предложенные программой варианты:

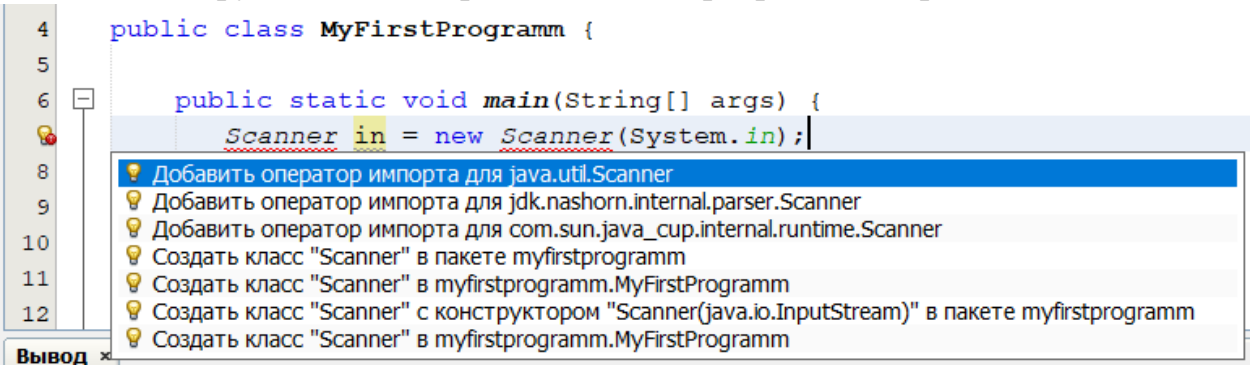


Рисунок 7 - Подсказка для импорта класса Scanner

Выберите первый вариант из списка (рис. 7).

Для создания самого объекта Scanner в его конструктор передается объект System.in. После этого мы можем получать вводимые значения.

Конструкция Scanner:

```
Scanner in = new Scanner(System.in);
```

```
int num = in.nextInt();
```

Чтобы получить введенное число, используется метод `in.nextInt();`, который возвращает введенное с клавиатуры целочисленное значение.

Пример программы:

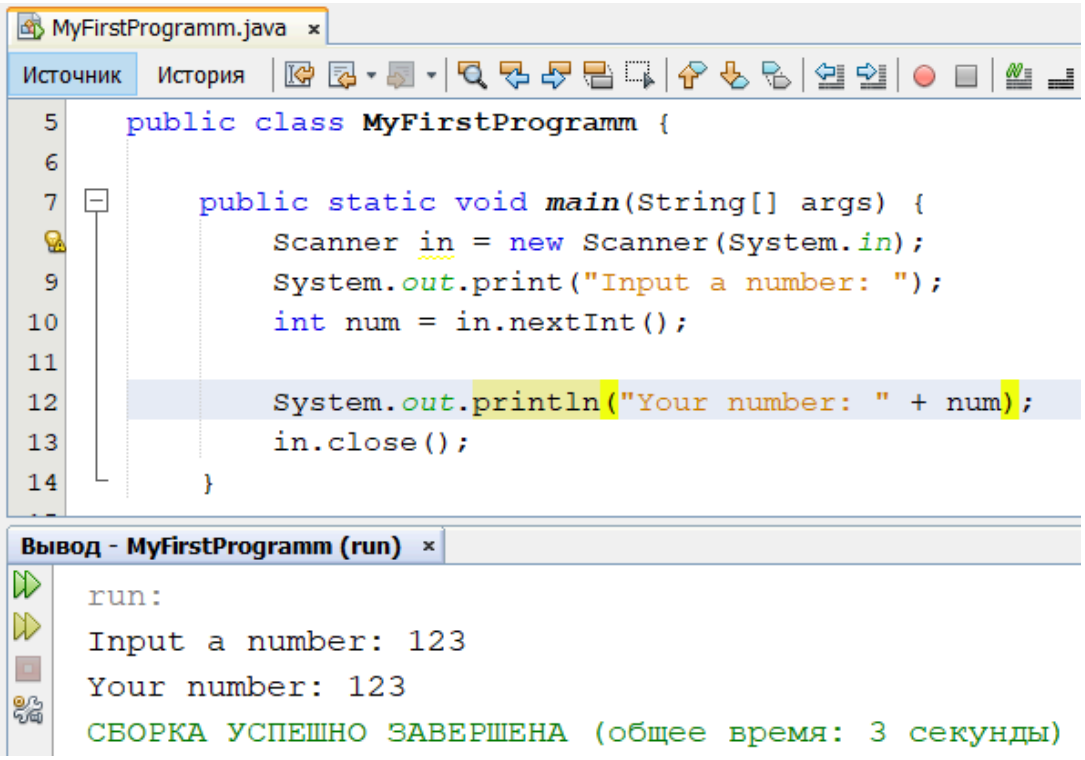


Рисунок 8 - Пример программы

Класс Scanner имеет ряд методов, которые позволяют получить введенные пользователем значения:

- **next()**: считывает введенную строку до первого пробела
- **nextLine()**: считывает всю введенную строку
- **nextInt()**: считывает введенное целое число **int**
- **nextDouble()**: считывает введенное вещественное число **double**
- **nextFloat()**: считывает введенное вещественное число **float**
- **nextBoolean()**: считывает значение **boolean**
- **hasNext()**: проверяет, было ли введено слово
- **hasNextInt()**: проверяет, было ли введено число **int**
- **hasNextDouble()**: проверяет, было ли введено **double**
- и др.

Класс MATH и математические методы

Класс Math содержит методы для выполнения основных числовых операций, таких как нахождение минимального значения, квадратного корня, среднего значения и т. д.

В классе Math имеются две константы типа double: **E** и **PI**. Все методы в классе Math статичны.

Основные математические методы:

Метод	Тип	Пример	Описание
max(int a, int b)	int	Math.max(7,2)=7	Возвращает больший из аргументов.
min(int a, int b)	int	Math.min(7,2)=2	Возвращает меньший из аргументов.
sqrt(double a)	double	Math.sqrt(64)=8	Возвращает квадратный корень аргумента.
cos(double a)	double	Math.cos(45)=0.5253	Возвращает косинус аргумента.
sin(double a)	double	Math.sin(45)=0.8509	Возвращает синус аргумента.
tan(double a)	double	Math.tan(45)=1	Возвращает тангенс аргумента.
abs (double a)	double	Math.abs(-6.3)=6.3	Возвращает абсолютное значение (модуль) числа типа double.
exp(double a)	double	Math.exp(3)= 20.0855	Возвращает основание натурального логарифма, возведенное в степень a.
log(double a)	double	Math.log(10)= 1	Возвращает натуральный логарифм (по основанию e).
log10(double a)	double	Math.log10(25)= 3.2188	Возвращает логарифм по основанию 10.
floorDiv(int a, int b)	int	Math.floorDiv(7, 2)=3	Возвращает целочисленный результат деления a на b.
pow(double a, double b)	double	Math. pow(2, 6)=64	Возвращает число a, возведенное в степень b.
PI	double	Math.PI= 3.141592653589793	Возвращает значение числа Пи.
random()	double	Math.random()	Возвращает случайное число от 0.0 (включительно) до 1 (не включительно).

		Math.random()*200 – 100	Возвращает случайное число от [-100;100].

ПРАКТИЧЕСКАЯ ЧАСТЬ

Задание 1: Написать первую программу, выводящую на окно вывода строку «Привет мир! Это моя первая программа!».

Задание 2: Проанализируйте следующий код метода. Что произойдет в результате его вызова, если было введено число 800?

```
public static void main (String args[]){
     int x;
     double y;
     x = 562;
     y = 8;
     Scanner s = new Scanner(System.in);
     x = s.nextInt();
     System.out.println(x/y);
}
```

Задание 3: Написать программу вычисления площади круга:

$$S = \pi R^2.$$

Задание 4: Написать программу вычисления площади кольца:

$$S = \pi \cdot (R^2 - r^2),$$

где **R** - внешний радиус кольца, **r** - внутренний радиус кольца.

Задание 5: Написать программу вычисления объема и площади поверхности цилиндра:

$$V = \pi R^2 h$$

$$S = 2\pi Rh + 2\pi R^2$$

Задание 6: Написать программу пересчета расстояния из верст в километры (1 верста равняется 1066,8 м).

Задание 7: Написать программу, которая преобразует введенное с клавиатуры дробное число в денежный формат. Например, число 125 должно быть преобразовано к виду 12 руб. 50 коп.

Примечание: *Используйте знак % - получение остатка от деления двух чисел (258 % 10; Результатом вычисления будет число 8).*

Задание 8: Написать программу пересчета величины временного интервала, заданного в минутах, в величину, выраженную в часах и минутах.

Задание 9: Написать программу для вычисления стоимости покупки, состоящей из нескольких тетрадей, карандашей и линеек.

Исходные данные: Цена тетради (руб.), Количество тетрадей, Цена карандаша (руб.), Количество карандашей, Цена линейки (руб.), Количество линеек, Стоимость покупки.

Задание 10: Проанализируйте следующий код метода. Что произойдет в результате его вызова?

```
public static void main (String args[]){
     int x, y;
     x = 5;
     y = x+2;
     x = (x+2)*(y-3);
     x = x % 5;
     y++;
     y = (x+14) % 7;
     System.out.println("x = " + x + " y = " + y);
}
```

Лабораторная работа № 2

«Логические и условные операторы на ЯП Java (IF и CASE)»

Цель: освоить логический оператор **if** … **else** и оператор повтора **case**.

Общие компетенции:

ОК 2. Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.

ОК 3. Принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность.

ОК 4. Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.

ОК 8. Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации.

Профессиональные компетенции:

ПК 2.2. Разрабатывать и публиковать программное обеспечение и информационные ресурсы отраслевой направленности со статическим и динамическим контентом на основе готовых спецификаций и стандартов.

ПК 2.3. Проводить отладку и тестирование программного обеспечения отраслевой направленности.

ПК 2.4. Проводить адаптацию отраслевого программного обеспечения.

ПК 2.5. Разрабатывать и вести проектную и техническую документацию.

ПК 2.6. Участвовать в измерении и контроле качества продуктов.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Оператор if/else

Общая форма условного оператора **if** в Java такая:

```
if (условие) {
   //действие(-я), которые выполняются, если условие
   истинно;
}
else if (условие) {
   //действие(-я), которые выполняются, если условие
истинно;
} ...
```

Выражение **if/else** проверяет истинность некоторого условия и в зависимости от результатов проверки выполняет последующее выражение или блок выражений после фигурных скобок.

Например:

```
int number1 = 2;
int number2 = 7;
if(number1 < number2){
```

```
    System.out.println("Первое число меньше второго");
}
```

Так как, в данном случае первое число меньше второго, то выражение **number1 < number2** истинно и возвращает значение **true**. Следовательно, управление переходит в блок кода после фигурных скобок и начинают выполняться содержащиеся там инструкции, а конкретно метод `System.out.println("Первое число больше второго");`.

Если бы первое число оказалось больше второго или равно ему, то инструкции в блоке **if** не выполнялись бы. В таком случае применяется блок **else**. Например:

```
int number1 = 2;
int number2 = 7;
if(number1 > number2){
    System.out.println("Первое число больше второго");
}
else{
    System.out.println("Первое число меньше второго");
```

Операторы сравнения в операторе **if**:

- **==** - определяет, что два числа равны друг другу;
- **!=** - определяет, что два числа не равны друг другу;
- **>** - определяет, больше ли одно число другого;
- **>=** - определяет, что одно число больше либо равно другого числа;
- **<** - определяет, меньше ли одно число другого;
- **<=** - определяет, что одно число меньше либо равно другого числа.

Логические операции в операторе **if**:

- **!** - оператор отрицания;
- **&&** - оператор логическое И (сокращенный);
- **||** - оператор логическое ИЛИ (сокращенный);
- **&** - оператор побитовое И;
- **|** - оператор побитовое ИЛИ.

<u>Примечание:</u> *Операторы сравнения и Логические операции необходимы для составления условия(ий) в операторе* ***if*** *!*

Оператор case

Общая форма конструкции **switch/case** в Java такая:

```
switch(ВыражениеДляСравнения) {
     case Совпадение1:
    команда;
     break;
     case Совпадение2:
```

```
    команда;
     break;
case Совпадение3:
    команда;
     break;
default:
    оператор;
    break;
}
```

Конструкция switch/case аналогична конструкции if/else, так как позволяет обработать сразу несколько условий.

Например:

```
int number = 7;
switch(number){
    case 1:
        System.out.println("число равно 1");
        break;
    case 7:
        System.out.println("число равно 7");
        num++;
        break;
    case 9:
        System.out.println("число равно 9");
        break;
    default:
        System.out.println("число не равно 1, 7, 9");
}
```

После ключевого слова **switch** в скобках идет сравниваемое выражение. Значение этого выражения последовательно сравнивается со значениями, помещенными после оператора **case**. И если совпадение будет найдено, то будет выполняться определенный блок **case**, если совпадение не найдено, то добавляется блок **default**, но он не обязателен.

В конце блока **case** ставится оператор **break**, чтобы избежать выполнения других блоков.

ПРАКТИЧЕСКАЯ ЧАСТЬ

Задание 1: Написать программу вычисления стоимости покупки с учетом скидки. Скидка в 3% предоставляется в том случае, если сумма покупки больше или равна 500 руб., в 5% — если сумма больше или равна 1000 руб.

```
Вычисление стоимости покупки с учетом скидки.
Введите сумму покупки и нажмите <Enter>
-> 640
Вам предоставляется скидка 3%
Сумма покупки с учетом скидки: 620.80 руб.
```

Задание 2: Написать программу, которая определяет четность или нечетность числа.

```
Введите число:                    Введите число:
-> 7                              -> 4
Число нечетное.                   Число четное.
```

Примечание: *Для того, чтобы определить четность числа следует в операторе if поставить условие:*

`if(a%2==0) {число четное},` *где a – введенное целое число.*

О знаке % прочтите примечание в лабораторной работе №1.

Задание 3: Написать программу решения квадратного уравнения:

$$\mathbf{ax^2 + bx + c = 0}.$$

Программа должна проверять правильность исходных данных и в случае, когда коэффициент при второй степени неизвестного равен нулю (a≠0), выводить соответствующее сообщение.

```
Введите в одной строке через пробел значения коэффициентов a,b и c, затем нажмите <Enter>
-> 12 27 -10
Корни уравнения:
x1= -25.551
x2= -28.449
-----------------------------------------------------------------
Введите в одной строке значения коэффициентов a,b и c, затем нажмите <Enter>
-> 0 27 -10
Ошибка! Коэффициент a не должен равняться нулю!
```

Задание 4: Написать программу, которая будет считывать введенное число и если это число четное и кратно 7, то вывести на экран красным строку «Этот текст красный», иначе вывести на экран зеленым строку «Этот текст зеленый».

Примечание: *Для того, чтобы вывести текст в зеленом и красном цветах, следует перед методом* ***main*** *написать следующие 2 строки кода:*

```
public static final String ANSI_RED = "\u001B[31m";
public static final String ANSI_GREEN = "\u001B[32m";.
```

В операторе ***if/else*** *в* ***System.out.println*** *перед текстом напишите имя соответствующей переменной, например* `System.out.println(ANSI_RED +"Этот текст красный");`.

```
Введите число, затем нажмите <Enter>
-> 70
Этот текст красный!
```

Задание 5: Написать программу, которая запрашивает у пользователя номер месяца и выводит соответствующее название времени года. В случае, если пользователь укажет недопустимое число, программа должна вывести сообщение "Ошибка ввода данных".

```
Введите номер месяца (число от 1 до 12) и нажмите <Enter>
-> 11
Ответ: Ноябрь – последний месяц осени.
```

Задание 6: Ввести значение, представляющее день недели. Программа должна определить название этого дня.

Задание 7: «Калькулятор». Ввести с клавиатуры числа и символ арифметического действия (+,-,*,/). Компьютер должен напечатать результат.

Введите через пробел два числа и символ арифметического действия, нажмите <Enter>
-> 5 8 +
Ответ: 13.
Введите через пробел два числа и символ арифметического действия, нажмите <Enter>
*-> 9 9 **
Ответ: 81.

Лабораторная работа № 3

«Логические и условные операторы на ЯП Java (WHILE и FOR)»

Цель: освоить циклы **while**, **do while** и **for**.

Общие компетенции:

ОК 1. Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.

ОК 4. Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.

ОК 5. Использовать информационно-коммуникационные технологии в профессиональной деятельности.

ОК 8. Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации.

Профессиональные компетенции:

ПК 2.2. Разрабатывать и публиковать программное обеспечение и информационные ресурсы отраслевой направленности со статическим и динамическим контентом на основе готовых спецификаций и стандартов.

ПК 2.3. Проводить отладку и тестирование программного обеспечения отраслевой направленности.

ПК 2.4. Проводить адаптацию отраслевого программного обеспечения.

ПК 2.5. Разрабатывать и вести проектную и техническую документацию.

ПК 2.6. Участвовать в измерении и контроле качества продуктов.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Цикл do/while

Конструкция оператора do while:

```
do {
    Тело цикла;
} while (условие выполнения);
```

Цикл **do** сначала выполняет код цикла, а потом проверяет условие в инструкции **while**. И пока это условие истинно, цикл повторяется. Например:

```
int j = 7;
do{
    System.out.println(j);
    j--;
} while (j > 0);
```

В данном случае код цикла сработает 7 раз, пока j не окажется равным нулю. Важно отметить, что цикл do гарантирует хотя бы однократное выполнение действий, даже если условие в инструкции while не будет истинно. Так, мы можем написать:

```
int j = -1;
do{
    System.out.println(j);
    j--;
} while (j > 0);
```

Хотя переменная j изначально меньше 0, цикл все равно один раз выполнится.

Цикл while

Конструкция оператора while:

```
while (Условие выполнения) {
    Тело цикла;
}
```

Цикл **while** сразу проверяет истинность некоторого условия, и если условие истинно, то код цикла выполняется:

```
int j = 6;
while (j > 0){
    System.out.println(j);
    j--;
}
```

Цикл for

Цикл for имеет следующее формальное определение:

```
for ([инициализация счетчика]; [условие]; [изменение счетчика]){
    // действия
}
```

Рассмотрим стандартный цикл for:

```
for (int i = 1; i < 9; i++){
    System.out.printf("Квадрат числа %d равен %d \n", i, i * i);
}
```

- Первая часть объявления цикла - `int i = 1` создает и инициализирует счетчик **i**. Счетчик необязательно должен представлять тип int, он может представлять другой числовой тип, например, float. Перед выполнением цикла значение счетчика будет равно 1. В данном случае это то же самое, что и объявление переменной.
- Вторая часть - условие, при котором будет выполняться цикл. В данном случае цикл будет выполняться, пока **i** не достигнет **9**.
- И третья часть - приращение счетчика на единицу, но необязательно увеличивать на единицу, можно уменьшать: i--.

В итоге блок цикла сработает 8 раз, пока значение i не станет равным 9.

И каждый раз это значение будет увеличиваться на 1.

ПРАКТИЧЕСКАЯ ЧАСТЬ

Задание 1: С помощью цикла **while** написать программу, вычисляющую сумму последовательности положительных чисел, которые вводятся с клавиатуры.

```
Для завершения работы программы введите 0 (ноль)
-> 45
-> 23
-> 15
-> 0
Сумма чисел: 83
```

Задание 2: С помощью цикла **while** вывести все квадраты натуральных чисел, не превосходящие данного числа N.

```
-> Введите n: 50
Квадраты натуральных чисел: 1 4 9 16 25 36 49
```

Задание 3: С помощью цикла **while** написать программу вычисляющую факториал числа N.

Формула факториала: **N! = 1*2*3*…*N**

```
-> Введите число: 3
Факториал числа 3 равен 6
```

Задание 4: С помощью цикла **do while** определить сумму цифр введенного числа.

```
-> Введите число: 455
Сумма цифр = 14.
```

Задание 5: С помощью цикла **do while** написать программу, которая определяет максимальное число из введенной с клавиатуры последовательности положительных чисел (длина последовательности не ограничена).

```
Для завершения работы программы введите 0 (ноль)
-> 56
-> 75
-> 43
-> 0
Максимальное число: 75
```

Задание 6: С помощью цикла **for** необходимо вывести на экран числа от 1 до 15 и наоборот.

Задание 7: С помощью цикла **for** вычислить сумму элементов фрагмента последовательности 2, 4, 6, 8, …, 98, 100.

Задание 8: С помощью цикла **for** написать программу, которая определяет минимальное и максимальное числа из введенной с клавиатуры последовательности положительных чисел (длина последовательности задается с консоли).

```
Введите длину последовательности:
-> 3
Введите 3 числа:
-> 56
-> 75
-> 43
Минимальное число: 43.
Максимальное число: 75.
```

Задание 9: С помощью цикла **for** написать программу, которая выводит таблицу значений функции

$$y = -2{,}4x^2 + 5x - 3$$

в диапазоне от -2 до 2 с шагом 0,5.

X	Y
-2	-22.60
-1.5	-15.90
-1	-10.40
-0.5	-6.10
0	-3.00
0.5	-1.10
1	-0.40
1.5	-0.90
2	-2.60

Лабораторная работа № 4

«Массивы в ЯП Java. Сортировка массивов в ЯП Java»

Цель: освоить одномерные и многомерные массивы, научиться производить сортировку массивов.

Общие компетенции:

ОК 1. Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.

ОК 4. Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.

ОК 5. Использовать информационно-коммуникационные технологии в профессиональной деятельности.

ОК 8. Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации.

Профессиональные компетенции:

ПК 2.2. Разрабатывать и публиковать программное обеспечение и информационные ресурсы отраслевой направленности со статическим и динамическим контентом на основе готовых спецификаций и стандартов.

ПК 2.3. Проводить отладку и тестирование программного обеспечения отраслевой направленности.

ПК 2.4. Проводить адаптацию отраслевого программного обеспечения.

ПК 2.5. Разрабатывать и вести проектную и техническую документацию.

ПК 2.6. Участвовать в измерении и контроле качества продуктов.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Одномерный массив

Создание массива:

```
int nums[] = new int[4];     // массив из 4 чисел
int[] nums2 = new int[5];    // массив из 5 чисел
```

Заполнение массива с консоли:

```
Scanner s = new Scanner(System.in);
int n = s.nextInt();    //Задается размерность массива
int []arr = new int[n];
for (int i = 0; i < arr.length; i++) {
     arr[i] = s.nextInt();   //Заполнение массива числами с
                               консоли
}
for (int i = 0; i < arr.length; i++) { //Вывод массива на
                                        консоль в строку
                                        через пробел
```

```
        System.out.print(arr[i] + " ");
    }
```

Двумерный массив

Создание массива:

```
    int nums[][] = new int[4][3]; // массив из 4 строк и 3 столбцов
    int[][] nums2 = new int[2][3]; //массив из 2 строк и 3 столбцов
```

Заполнение массива с консоли:

```
Scanner s = new Scanner(System.in);
int m = s.nextInt();    //Задается размерность строк
int n = s.nextInt();    //Задается размерность столбцов
int [][]arr = new int[m][n];
for (int i = 0; i < arr.length; i++) {
for (int j = 0; j < arr[i].length; j++) {
          arr[i][j] = s.nextInt(); //Заполнение массива
                                   числами с консоли
}
}
for (int i = 0; i < arr.length; i++) {
for (int j = 0; j < arr[i].length; j++) {
     System.out.print(arr[i][j] + "\t"); //Вывод массива
                                         на консоль
}
System.out.println();
 }
```

ПРАКТИЧЕСКАЯ ЧАСТЬ

Задание 1: Дан одномерный массив целых чисел {3, 5, 8, 12, 11, 1, 8}. На консоль вывести среднее арифметическое элементов массива.

Задание 2: Ввести с консоли n целых чисел и поместить их в одномерный массив. На консоль вывести четные и нечетные числа.

Задание 3: Ввести с консоли n целых чисел и поместить их в одномерный массив. На консоль вывести **максимум** и **минимум**.

Задание 4: Создайте одномерный массив из всех чётных чисел от 2 до 20 и выведите элементы массива на экран.

Задание 5: Вывести на экран матрицу **m*n** (где **m** - количество строк, а **n** - количество элементов в строке (столбцы)) вида:

1	2	3	4
5	6	7	8
9	10	11	12

Задание 6: Дан целочисленный двумерный массив, размерности **m*n**, найти сумму элементов массива в каждой строке.

Задание 7: Дан целочисленный двумерный массив, размерности **m*n.** Заменить все элементы на их квадраты.

Лабораторная работа № 5

«Создание классов на ЯП Java. Конструкторы, методы и поля классов»

Цель: получение навыков работы с классами, предназначенными для хранения разнородной информации о реальных объектах.

Общие компетенции:

ОК 1. Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.

ОК 4. Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.

ОК 5. Использовать информационно-коммуникационные технологии в профессиональной деятельности.

ОК 8. Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации.

Профессиональные компетенции:

ПК 2.2. Разрабатывать и публиковать программное обеспечение и информационные ресурсы отраслевой направленности со статическим и динамическим контентом на основе готовых спецификаций и стандартов.

ПК 2.3. Проводить отладку и тестирование программного обеспечения отраслевой направленности.

ПК 2.4. Проводить адаптацию отраслевого программного обеспечения.

ПК 2.5. Разрабатывать и вести проектную и техническую документацию.

ПК 2.6. Участвовать в измерении и контроле качества продуктов.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Java - это объектно-ориентированный язык программирования. Все программы состоят из объектов, которые как-то связываются между собой.

Класс - это, по сути, шаблон для объекта. Он определяет, как объект будет выглядеть и какими функциями обладать. Каждый объект является объектом какого-то класса.

Модификаторы доступа:

- **public** определяет, что следующие за ним определения доступны всем;

- **private** означает, что следующие за ним определения может использовать только создатель типа, внутри функций членов этого типа;

Модификатор	Класс	Пакет	Класс-потомок	Все классы
public	+	+	+	+
protected	+	+	+	–
нет	+	+	–	–
private	+	–	–	–

- **protected** по действию схож с private, за одним исключением: унаследованные классы имеют доступ к членам, помеченным protected, хотя и не имеют доступа к private-членам.

ПРАКТИЧЕСКАЯ ЧАСТЬ

Допустим перед нами стоит задача создать класс, который будет описывать характеристики и поведение кота.

Структура любого класса в java

В предыдущих лабораторных работах при создании проекта автоматически создавался класс одного вида:

```
public class Cat {
    // поля, конструкторы и методы
    //тело класса
}
```

Рассмотрим составляющие класса поподробнее:

- **public** – модификатор доступа к классу, в данном случае он нам говорит, что этот класс будет доступен не только данному классу, но и другим.
- **class** – ключевое слово, говорящее о том, что это класс.
- **Cat** – имя класса. Имена классов принято писать с заглавной буквы.
- **{ }** – фигурные скобки, между которыми разместится тело нашего класса.

Поля класса Cat

Атрибутами кота могут быть: *имя, вес, окраска*. Поля (атрибуты) это переменные, которые объявляются следующим образом:

```
public int weight; // вес кота
public String name; // имя кота
public String color; //окрас кота
```

int и String – это типы данных. В данном случае вес будет задан при помощи целого числа – `int`, а имя и цвет при помощи символьной строки `String`.

Примечание: После объявления каждого атрибута должна ставиться точка с запятой (**;**).

Методы класса Cat

Пусть наш кот умеет есть, спать и разговаривать. Опишем это поведение с помощью методов:

```
//кот ест
 public void eat(){
      System.out.print("Eating...\n");
 }
//кот спит
 public void sleep(){
      System.out.print("Sleeping zz-z-z-z...\n");
 }
//кот говорит
public String speak(String words){
      String phrase = words + "...mauu...\n";
      return phrase;
}
```

Метод **speak(String words)** в отличие от предыдущих методов возвращает значение и имеет входные параметры. Давайте подробнее рассмотрим сигнатуру метода public String speak(String words):

- **public String** - доступен для других классов и тип значения, которое возвращает метод. В предыдущих случаях ключевое слово **void** указывало на то, что метод ничего не возвращает. В данном случае String указывает на то, что метод возвращает значение типа строка.

Что же это значит? В процессе своей работы метод выполняет определенные действия над данными. Иногда необходимо, чтобы результат этих действий был передан для дальнейшей обработки другим классам, в этом случае метод передает (возвращает) этот результат. Эти возвращаемые данные относятся к какому-либо типу. В нашем примере это тип символьной строки, String.

Возвращающие методы должны содержать в своем теле ключевое слово **return**, которое указывает на то, что именно возвращает данный метод. В нашем случае это переменная phrase.

- **(String words)** - входные параметры. **Входные параметры** - это какие-либо данные, которые передаются из других классов и, которые метод должен обработать. Наш метод получает в качестве входных данных строку в виде переменной words, к этой строке дописывает «...mauu...» и возвращает то, что получилось.

В итоге класс Cat выглядит следующим образом:

```
1   package cat;
2
3   public class Cat {
4
5       public int weight; // вес кота
6       public String name; // имя кота
7       public String color; //окрас кота
8
9       //кот ест
10      public void eat(){
11          System.out.print("Eating...\n");
12      }
13
14      //кот спит
15      public void sleep(){
16          System.out.print("Sleeping zz-z-z-z...\n");
17      }
18
19      //кот говорит
20      public String speak(String words){
21          String phrase = words + "...mauu...\n";
22          return phrase;
23      }
24
25  }
```

Рисунок 9 – Листинг класса Cat

Обращение к классу в Java

Пусть в нашем проекте есть еще один класс под названием MainClass.

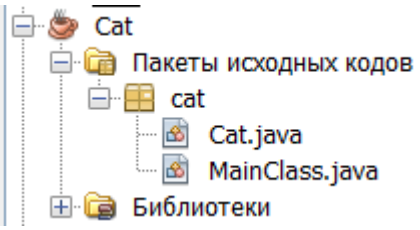


Рисунок 10 – Дерево проекта

Его конструкция выглядит так:

```
public class MainClass {

public static void main(String[] args) {

}
}
```

Прежде чем вызывать созданные нами методы в классе Cat и заставить нашего кота есть, спать и говорить, сперва нужно создать экземпляр класса (инстанцию).

```
Cat vasia = new Cat();
```

Данная строчка нам говорит о том, что в памяти создан экземпляр объекта Cat, а переменная vasia типа Cat (такого же, как и наш объект) указывает на то место в памяти, где был этот объект создан.

Переменную vasia теперь можно использовать для вызова методов класса Cat, например:

```
vasia.eat();
vasia.sleep();
```

При вызове этих методов в программе NetBeans удобно пользоваться комбинацией клавиш **Ctrl + пробел**, после введения имени переменной и точки. Программа подскажет, какие можно использовать методы для данной переменной.

Если метод возвращает какое-либо значение, например, как наш метод speak() возвращает значение типа String, то его можно вызывать следующим образом:

- объявить переменную такого же типа, что и возвращаемое значение (в нашем случае String)

- присвоить ей вызванный метод, например:

```
String say = vasia.speak("Play with me");
```

Вспомним, что при описании нашего метода он содержал параметры speak(String words). Теперь, при вызове в качестве параметра выступила фраза "Play with me", метод speak() ее обработал и вернул "Play with me...mauu...". Именно это значение он присвоил переменной say.

Мы это можем проверить, выведя say на печать при помощи команды:

```
System.out.println(say);
```

Итак, наш класс MainClass теперь выглядит следующим образом:

```
1   package cat;
2
3   public class MainClass {
4       public static void main(String[] args) {
5
6           Cat vasia = new Cat();
7           vasia.eat();
8           vasia.sleep();
9           String say = vasia.speak("Play with me");
10          System.out.println(say);
11          }
12  }
```

Рисунок 11 – Листинг класса MainClass

В качестве результата внизу в консоли мы получаем следующие строки:

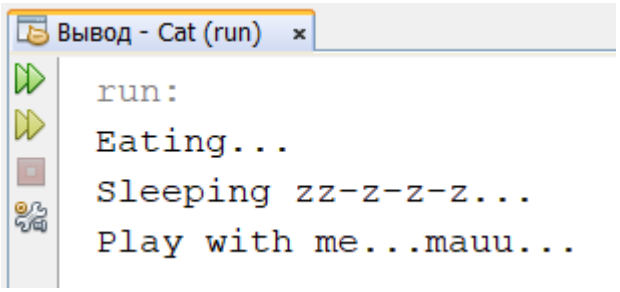


Рисунок 12 – Окно вывода проекта Cat

Мы описали все поведения кота, но конкретно какого кота? Мы назвали объект vasia, но это не означает, что поле **name**, созданное в классе **Cat**, будет присваивать название vasia. Давайте теперь дадим нашему коту имя, скажем какой у него вес и окрас:

```java
package cat;

public class MainClass {
    public static void main(String[] args) {

    Cat vasia = new Cat();

    vasia.name = "Васька";
    vasia.weight = 10;
    vasia.color = "Рыжий";

    vasia.eat();
    vasia.sleep();
    String say = vasia.speak("Play with me");
    System.out.println(say);

    }
}
```

Рисунок 13 – Листинг класса MainClass с добавлением имени и окраса

А что если дома не один кот? Это не проблема для нашего проекта. Ведь у нас уже описан сам объект типа Кот и все его характеристики и поля. Также как создавали кота Ваську, можно создать подобным образом сколько угодно котов. Например создадим еще кота и назовем его Барсик:

```java
package cat;

public class MainClass {
    public static void main(String[] args) {

        Cat vasia = new Cat();

        vasia.name = "Васька";
        vasia.weight = 10;
        vasia.color = "Рыжий";

        System.out.println("Кот " + vasia.name + ":");
        vasia.eat();
        vasia.sleep();
        String say = vasia.speak("Play with me");
        System.out.println(say);

        Cat barsik = new Cat();

        barsik.name = "Жирный Барсик";
        barsik.weight = 5;
        barsik.color = "Белый";

        System.out.println("Кот " + barsik.name + ":");
        barsik.eat();
        barsik.sleep();
        String sayBarsik = barsik.speak("Give me more food slave");
        System.out.println(sayBarsik);

    }
}
```

Рисунок 14 – Листинг класса MainClass с созданием еще одного кота

В качестве результата внизу в консоли мы получаем следующие строки:

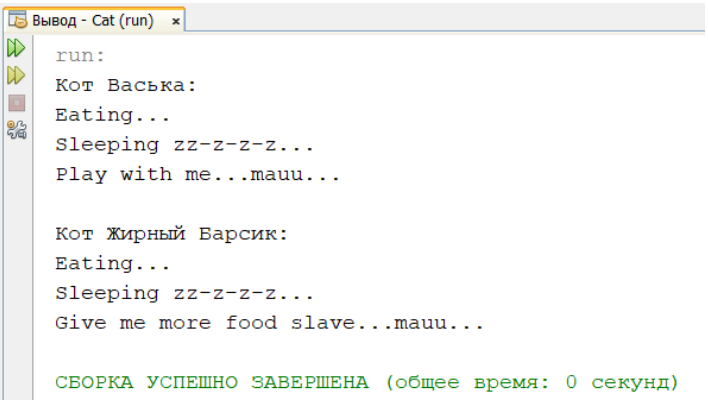


Рисунок 15 – Окно вывода проекта Cat

САМОСТОЯТЕЛЬНАЯ ЧАСТЬ

Задание 1: Данные типа «песня» (Song) описывают следующие свойства объекта:

• название песни (name);

• исполнитель (perfonner);

• продолжительность (length), в секундах.

Дана таблица для класса Song:

String name	Поля
String perfomer	
int length	
String getName()	Методы, возвращающие значения
String getPerfomer() менее 2-х минут	
int getLength()	
void setName(String n)	Методы, устанавливающие значения
void setPefomer(String p)	
void setLength(int len)	

1. Напишите класс для объектов указанного типа.

2. Напишите метод Category, который получает в качестве параметра объект Song и возвращает одно из следующих значений: short (если

продолжительность песни менее 2-х минут), long (если продолжительность песни более 4-х минут), medium (если продолжительность менее 4-х, но более 2-х минут).

Задание 2: Данные типа «дата» (Data) описывают следующие свойства объекта:

• день (day);

• месяц (mounth);

• год (year).

Дана таблица для класса Data:

private int day	Поля
private int month;	
private int year;	
int getDay()	Методы, возвращающие значения
int getMounth()	
int getYear()	
void setDay(int d)	Методы, устанавливающие значения
void setMounth(int m)	
void setYear(int y)	

1. Напишите класс для объектов указанного типа.

2. Напишите метод compareTo, который получает в качестве параметра объект Date и возвращает одно из следующих значений:

- **0** (ноль), если даты совпадают
- **-1**, если данная дата предшествует дате-параметру
- **1**, если дата-параметр предшествует данной дате.

Лабораторная работа № 6

«Ввод и вывод данных в Java. Классы InputStream и OutputStream»

Цели:

1. Изучить применение классов пакета java.io для организации ввода-вывода данных в приложениях на языке Java.
2. Освоить работу с текстовыми и бинарными файлами.

Общие компетенции:

ОК 1. Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.

ОК 4. Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.

ОК 5. Использовать информационно-коммуникационные технологии в профессиональной деятельности.

ОК 8. Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации.

Профессиональные компетенции:

ПК 2.2. Разрабатывать и публиковать программное обеспечение и информационные ресурсы отраслевой направленности со статическим и динамическим контентом на основе готовых спецификаций и стандартов.

ПК 2.3. Проводить отладку и тестирование программного обеспечения отраслевой направленности.

ПК 2.4. Проводить адаптацию отраслевого программного обеспечения.

ПК 2.5. Разрабатывать и вести проектную и техническую документацию.

ПК 2.6. Участвовать в измерении и контроле качества продуктов.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Отличительной чертой многих языков программирования является работа с файлами и потоками. В Java основной функционал работы с потоками сосредоточен в классах из пакета java.io.

Пакет java.io содержит почти все классы, которые вам могут понадобиться для ввода и вывода (ввода/вывода) в Java. Все эти потоки представляют собой источник ввода и назначение вывода. Ключевым

понятием здесь является понятие **потока**. Хотя понятие "поток" в программировании довольно перегружено и может обозначать множество различных концепций. В данном случае применительно к работе с файлами и вводом-выводом мы будем говорить о потоке (stream), как об абстракции, которая используется для чтения или записи информации (файлов, сокетов, текста консоли и т.д.).

Поток

Поток может быть определен как последовательность данных.

Есть два вида потоков:

- **InPutStream** - используется для чтения данных из источника.
- **OutPutStream** - используется для записи данных в пункт назначения.

Потоки байтов

Класс InputStream

Класс InputStream является базовым для всех классов, управляющих байтовыми потоками ввода. Рассмотрим его основные методы:

- int **available()**: метод возвращает количество непрочитанных (доступных) байт;
- void **close()**: закрывает поток – вызывается после окончания работы с потоком.

Объект выполняет служебные операции, связанные с закрытием файла на диске и т.д.

Из потока больше нельзя читать данные;

- int **read()**: метод читает один байт и возвращает его как результат. Результат расширяется до int, для красоты. Если все байты уже прочитаны, метод вернет -1;

- int **read(byte[] buffer)**: считывает блок байтов из потока в массив buffer , пока buffer не заполнится или не закончатся байты там, откуда он их читает. После чтения возвращает число считанных байтов. Если ни одного байта не было считано, то возвращается число -1;
- long **skip(long number)**: пропускает в потоке при чтении некоторое количество байт, которое равно number.

Класс OutputStream

Класс OutputStream является базовым классом для всех классов, которые работают с бинарными потоками записи. Свою функциональность он реализует через следующие методы:

- void **close()**: закрывает поток;
- void **flush()**: если есть данные, которые хранятся где-то внутри и еще не записаны, то они записываются;
- void **write(int b)**: метод записывает один байт информации. Тип int сужается до byte, лишняя часть просто отбрасывается;
- void **write(byte[] buffer)**: записывает в выходной поток массив байтов buffer;
- void **write(byte[] buffer, int offset, int length)**: записывает в выходной поток некоторое число байтов, равное length, из массива buffer, начиная со смещения offset, то есть с элемента buffer[offset].

Байтовые потоки

Потоки байтов Java используются для ввода и вывода 8-битных байтов. Хотя существует много классов, связанных с потоками байтов, но наиболее часто используемые классы - FileInputStream и FileOutputStream соответственно, предназначенными для работы с текстовыми файлами.

Пример записи строки в файл

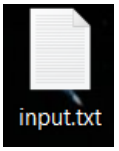
```java
package program;

import java.io.FileOutputStream;
import java.io.IOException;

public class Program {

    public static void main(String[] args) {
        String text = "Hello world!"; // строка для записи

        try(FileOutputStream fos=new FileOutputStream("C:\\Users\\User\\Desktop\\input.txt")) {
            // перевод строки в байты
            byte[] buffer = text.getBytes();

            fos.write(buffer, 0, buffer.length);
        }
        catch(IOException ex){

            System.out.println(ex.getMessage());
        }
        System.out.println("The file has been written");
    }

}
```

Рисунок 16 – Листинг класса Program

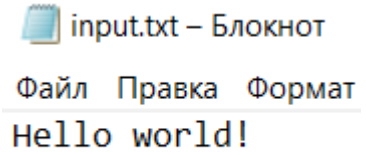
Для создания объекта FileOutputStream используется конструктор, принимающий в качестве параметра путь к файлу для записи (C:\\Users\\User\\Desktop\\input.txt). Если такого файла нет, то он автоматически создается при записи. Так как здесь записываем строку, то ее надо сначала перевести в массив байтов. И с помощью метода write строка записывается в файл.

Для автоматического закрытия файла и освобождения ресурса объект FileOutputStream создается с помощью конструктции **try...catch**.

Вот, что получилось:

На рабочем столе создался файл: 

.



Содержимое файла:

Пример чтения данных из файла

Для считывания данных из файла предназначен класс FileInputStream, который является наследником класса InputStream и поэтому реализует все его методы.

Для создания объекта FileInputStream мы можем использовать ряд конструкторов. Наиболее используемая версия конструктора в качестве параметра принимает путь к считываемому файлу:

```
FileInputStream(String fileName) throws FileNotFoundException
```

Если файл не может быть открыт, например, по указанному пути такого файла не существует, то генерируется исключение FileNotFoundException.

Считаем данные из ранее записанного файла и выведем на консоль:

```
1  package program;
2
3  import java.io.FileInputStream;
4  import java.io.IOException;
5
6  public class Program {
7
8      public static void main(String[] args) {
9          try(FileInputStream fin=new FileInputStream("C:\\Users\\User\\Desktop\\input.txt")) {
10             System.out.printf("File size: %d bytes \n", fin.available());
11
12             int i=-1;
13             while((i=fin.read())!=-1){
14                 System.out.print((char)i);
15             }
16         }
17         catch(IOException ex){
18             System.out.println(ex.getMessage());
19         }
20     }
21
22 }
```

Рисунок 17 – Считывание данных класса Program

В данном случае мы считываем каждый отдельный байт в переменную i:

```
while((i=fin.read())!=-1)
```

Когда в потоке больше нет данных для чтения, метод возвращает число -1.

Затем каждый считанный байт конвертируется в объект типа char и выводится на консоль.

Вот, что получилось:

Рисунок 17 – Считывание данных класса Program

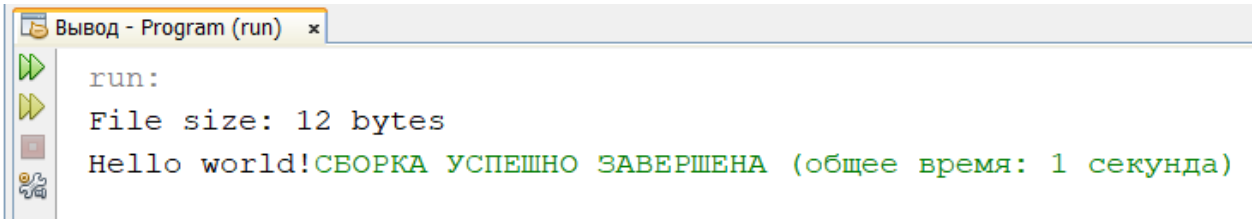


Рисунок 18 – Окно вывода класса Program

Чтение консольного ввода

Java не имеет обобщенного метода консольного ввода, который соответствует стандартной С-функции scanf() или операциям ввода.

Консольный ввод в Java выполняется с помощью считывания из объекта System.in. Чтобы получить символьный поток, который присоединен к консоли, нужно перенести ("упаковывать") System.in в объект типа BufferedReader.

Класс BufferedReader поддерживает *буферизированный входной поток*.

Пример чтения символов с консоли

```
1  package program;
2
3  import java.io.BufferedReader;
4  import java.io.IOException;
5  import java.io.InputStreamReader;
6
7  public class Program {
8
9      public static void main(String[] args) throws IOException {
10         char c;
11         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
12         System.out.println("Введите символы, 'q' - для завершения.");
13         // Чтение символов
14         do {
15             c = (char) br.read();
16             System.out.println(c);
17         } while(c != 'q');
18     }
19
20 }
```

Рисунок 19 – Чтение символов с консоли класса Program

Для чтения символа из BufferedReader используйте метод read(). Версия read(), которая была применена, такова:

```
int read() throws IOException
```

При каждом вызове read() читает символ из входного потока и возвращает его в виде целочисленного значения. Когда read() сталкивается с концом потока, то возвращает -1 Он может выбрасывать исключение ввода/вывода (I/O-исключение — IOException).

Вот, что получилось:

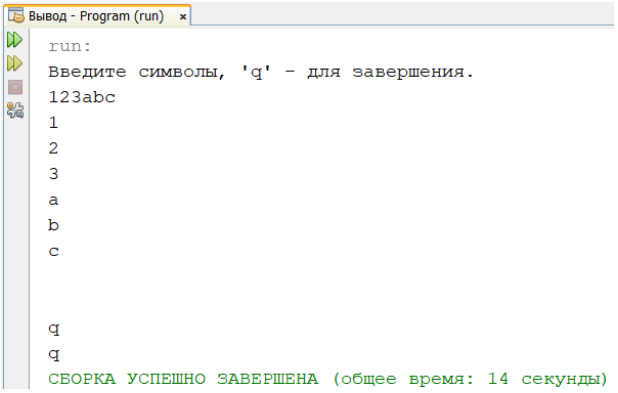


Рисунок 20 – Окно вывода класса Program

Пример чтения строк с консоли

```
3  import java.io.BufferedReader;
4  import java.io.IOException;
5  import java.io.InputStreamReader;
6
7  public class Program {
8
9      public static void main(String[] args) throws IOException {
10         String str;
11         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
12         System.out.println("Введите строку или строки, 'q' - для завершения.");
13         // Чтение символов
14         do {
15             str = br.readLine();
16             System.out.println(str);
17         } while(!str.equals("q"));
18     }
19
20 }
```

Рисунок 21 – Листинг чтения строк с консоли класса Program

Метод readLine() класса BufferedReader возвращает String-объект, который содержит строку, набранную на клавиатуре:

```
String readLine() throws IOException
```

Вот, что получилось:

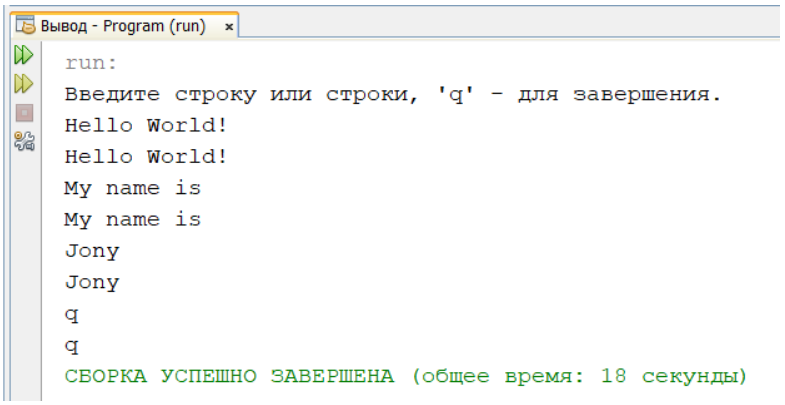


Рисунок 22 – Окно вывода класса Program

ПРАКТИЧЕСКАЯ ЧАСТЬ

Создайте код, показанный на рисунке ниже, в своем проекте:

```
3  import java.io.BufferedReader;
4  import java.io.FileOutputStream;
5  import java.io.IOException;
6  import java.io.InputStreamReader;
7
8  public class Program {
9
10     public static void main(String[] args) throws IOException {
11         String str;
12         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
13         System.out.println("Введите стих, 'q' - для завершения.");
14         // Чтение символов
15         try(FileOutputStream fos=new FileOutputStream("C:\\Users\\User\\Desktop\\input.txt")) {
16             do {
17                 str = br.readLine();
18                 str += "\n";
19                 // перевод строки в байты и зыпись в файл
20                 byte[] buffer = str.getBytes();
21                 fos.write(buffer, 0, buffer.length);
22             }while(!str.equals("q\n"));
23         }
24         catch(IOException ex){
25             System.out.println(ex.getMessage());
26         }
27
28     }
29
30 }
```

Рисунок 23 – Листинг класса Program

Запустите проект и вставьте ПОСТРОЧНО строки из **английского** стихотворения:

Sun of the sleepless! Melancholy star!	(Неспящих солнце, грустная звезда),
Whose tearful beam glows tremulously far!	(Как слезно луч мерцает твой всегда),
That shows the darkness thou canst not dispel,	(Как темнота при нем еще темней),
How like art thou to joy remembered well!	(Как он похож на радость прежних дней)!
So gleams the past, the light of other days,	(Так светит прошлое нам в жизненной ночи),
Which shines, but warms not with its powerless rays;	(Но уж не греют нас бессильные лучи),
A night beam Sorrow watched to behold,	(Звезда минувшего так в горе мне видна),
Distinct, but distant - clear - but, oh how cold!	(Видна, но далека - светла, но холодна)!

Созданный файл при открытии должен выглядеть так:

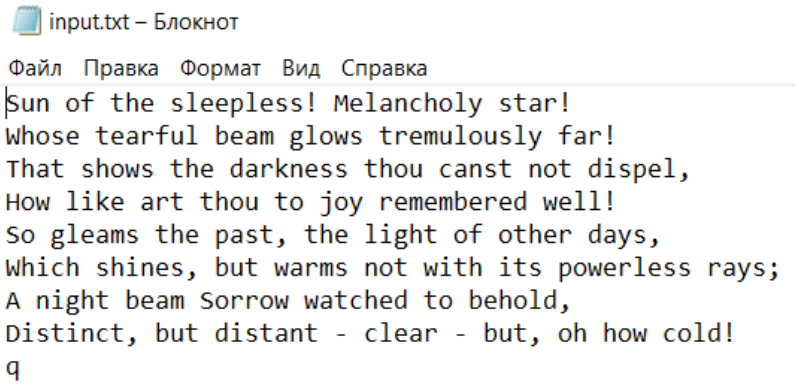


Рисунок 24 – Вывод в файл input.txt

Создали файл и заполнили его содержимым в виде текста. Теперь будем работать с ним. Для того, чтобы считать текст из файла понадобится смекалка.

Для начала измените весь код класса Program на следующий:

```java
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;

public class Program {

    public static void main(String[] args) throws IOException {
        File file = new File("C:\\Users\\User\\Desktop\\input.txt");
            //создаем объект FileReader для объекта File
            FileReader fr = new FileReader(file);
            //создаем BufferedReader с существующего FileReader для построчного считывания
            BufferedReader reader = new BufferedReader(fr);

            String line = reader.readLine(); // считаем сначала первую строку
            while (line != null) { // считываем остальные строки в цикле
                System.out.println(line);
                line = reader.readLine();
            }
        }
}
```

Рисунок 25 – Листинг класса Program

Если запустить проект, то в окне вывода можно увидеть то самое английское стихотворение.

Теперь, чтобы перейти к самостоятельной части нужно изменить код так, чтобы все строки были помещены в массив строк. Если их не поместить в массив, то работать далее будет очень затруднительно.

Измените код:

```java
public class Program {

    public static void main(String[] args) throws IOException {
        int count = 0;
        String str = "";
        String[] lines = new String[0];
        String[] buf;

        File file = new File("C:\\Users\\User\\Desktop\\input.txt");
        //создаем объект FileReader для объекта File
        FileReader fr = new FileReader(file);
        //создаем BufferedReader с существующего FileReader для построчного считывания
        BufferedReader reader = new BufferedReader(fr);

        String line = reader.readLine(); // считаем сначала первую строку
        while ((str = reader.readLine()) != null) { // считываем остальные строки в цикле
            buf = lines;
            lines = new String[count + 1];
            for (int i = 0; i < buf.length; i++) {
                lines[i] = buf[i];
            }
            lines[count] = str;
            count++;
        }
        for (int i = 0; i < lines.length; i++) {
            System.out.println(lines[i]);  //это массив строк необходимый для дальнейшего использования
        }
        reader.close();
        }
}
```

Рисунок 26 – Листинг класса Program

САМОСТОЯТЕЛЬНАЯ ЧАСТЬ:

Задание 1: В каждой строке определить самое длинное слово.

Длину строки можно определить через метод: `str.length();`.

Задание 2: Подсчитать сколько всего в стихотворении есть букв “ s ”, “ W ” и сколько знаков восклицания “ ! ”.

Для выполнения этого задания каждую строку из массива строк lines[] нужно преобразовать в массив символов: `char[] ch = lines[i].toCharArray();`. Далее пройтись по этому массиву символов ch[] и подсчитать количество символов.

Примечание: При переходе на новую строку из массива lines[] лучше пересоздавать массив символов ch[].

Лабораторная работа № 7,8

«Использование визуального редактора GUI в NetBeans. Создание калькулятора посредством форм»

Цель: освоить одномерные и двумерные массивы.

Общие компетенции:

ОК 1. Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.

ОК 4. Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.

ОК 5. Использовать информационно-коммуникационные технологии в профессиональной деятельности.

ОК 8. Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации.

Профессиональные компетенции:

ПК 2.2. Разрабатывать и публиковать программное обеспечение и информационные ресурсы отраслевой направленности со статическим и динамическим контентом на основе готовых спецификаций и стандартов.

ПК 2.3. Проводить отладку и тестирование программного обеспечения отраслевой направленности.

ПК 2.4. Проводить адаптацию отраслевого программного обеспечения.

ПК 2.5. Разрабатывать и вести проектную и техническую документацию.

ПК 2.6. Участвовать в измерении и контроле качества продуктов.

ПРАКТИЧЕСКАЯ ЧАСТЬ

1. Создайте проект и назовите его Calculator
2. В дереве проектов нажмите на пакет с именем calculator правой кнопкой мыши и выберете Новый – Форма JFrame:

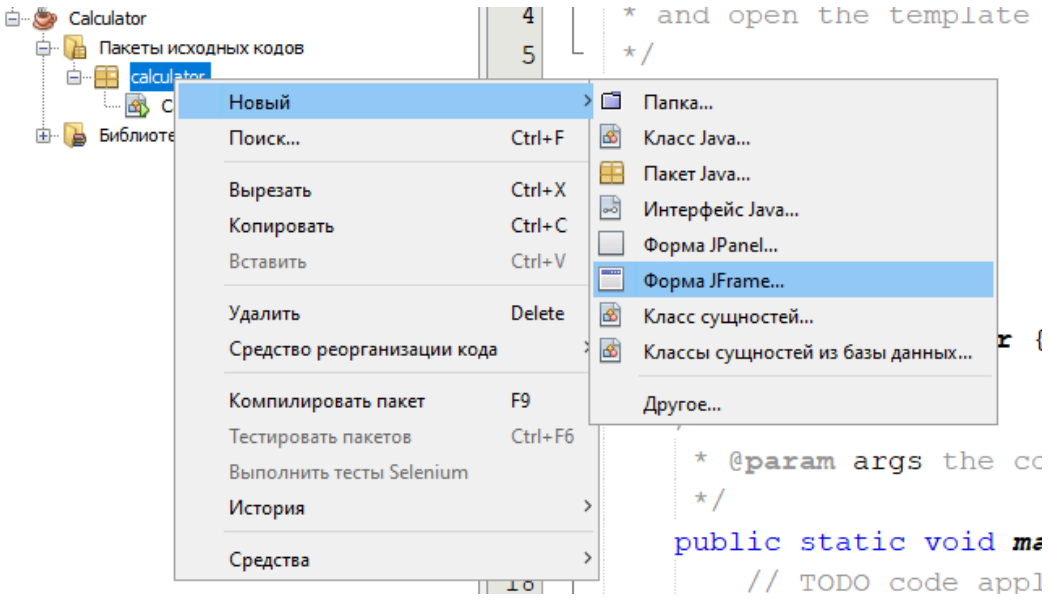


Рисунок 27 – Меню создания новой формы

3. В открывшемся окне дайте название форме «CalculatorFrame» и нажмите кнопку Готово:

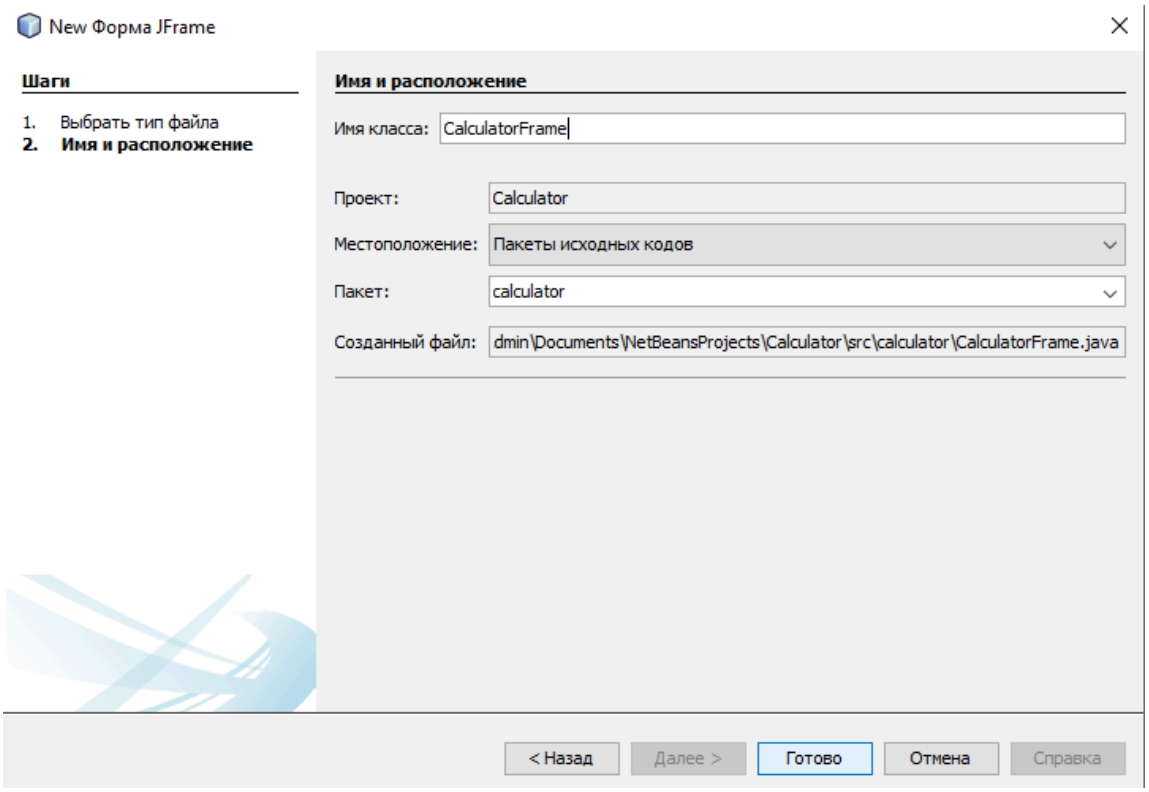


Рисунок 28 – Окно создания новой формы

4. Созданный Frame будет выглядеть следующим образом:

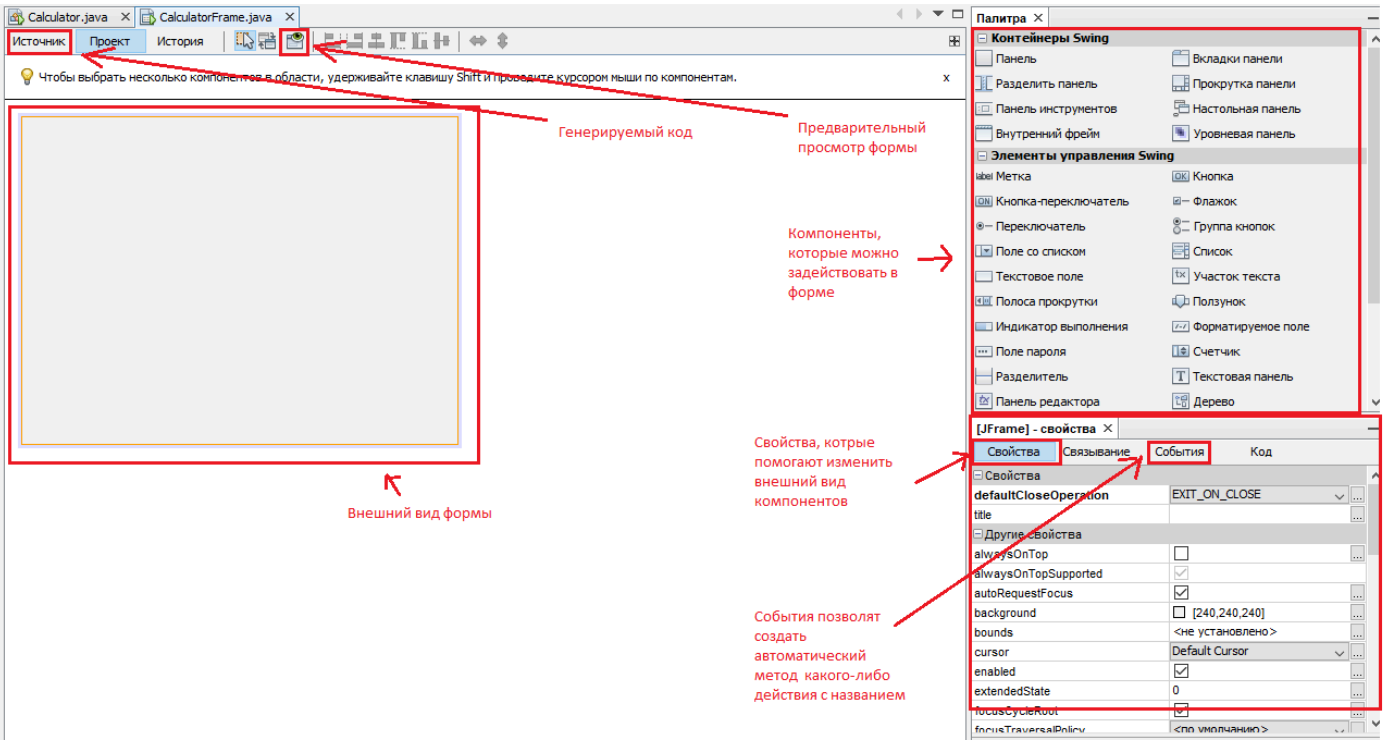


Рисунок 29 – Окно новой формы

5. Растяните форму до размеров 350x400.

6. Для того, чтобы начать заполнять форму сначала нужно добавить на нее «Панель». Перетащите Панель из Палитры компонентов и растяните на всю область формы и примените цвет Белый.

Рисунок 30 – Добавление Панели в форму

Чтобы изменить цвет панели нужно в окне Свойств найти background и нажать у него кнопку с троеточием. Откроется окно в палитрой цветов.

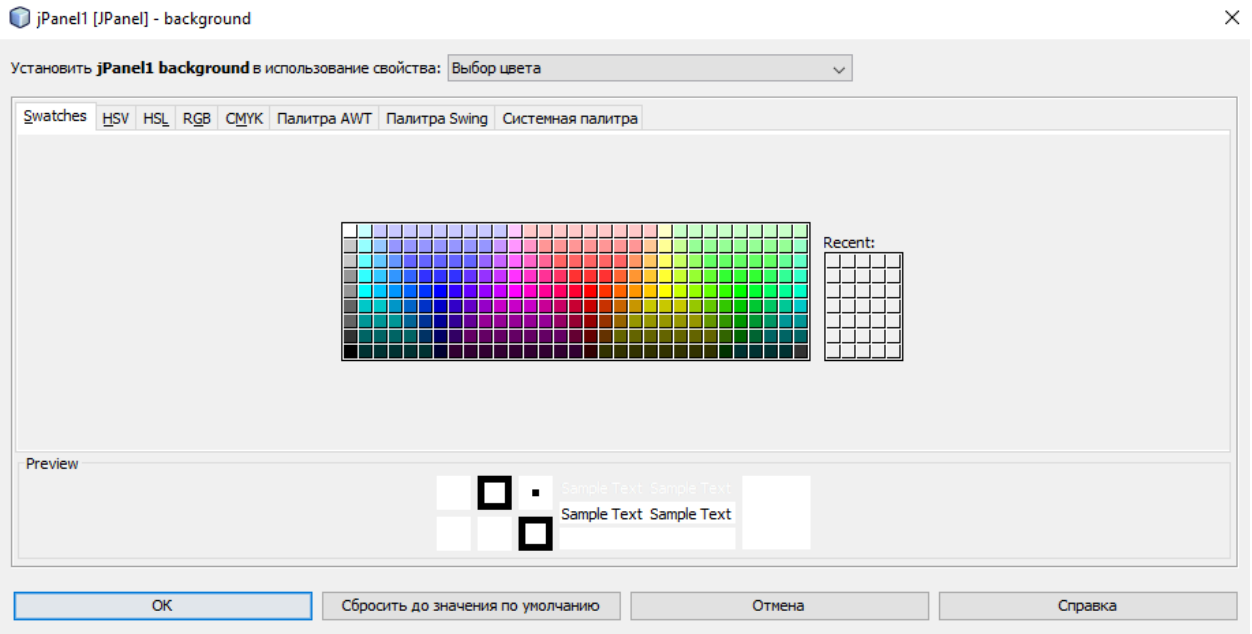


Рисунок 31 – Палитра цветов

7. Теперь разместите все компоненты как показано на рисунке

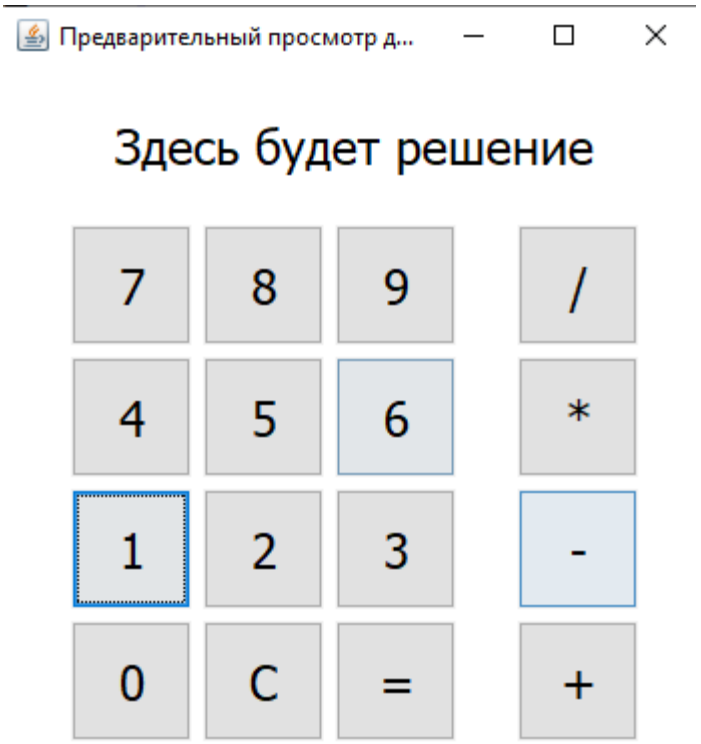


Рисунок 32 – Итоговый вид формы

Свойства каждой кнопки:

- Размер текста – 24
- Размер кнопки 60x60
- focusable снять галочку.

Для будущего решения применяется компонент JLable – Метка.

8. Для того, чтобы кнопки заработали нужно установить слушатели. Для этого перейдите на вкладку События, далее найдите метод mouseClicked и нажмите кнопку с троеточием. В открывшемся окне нажмите кнопку добавить, дайте имя кнопке - clicked1 и нажмите кнопку ОК два раза:

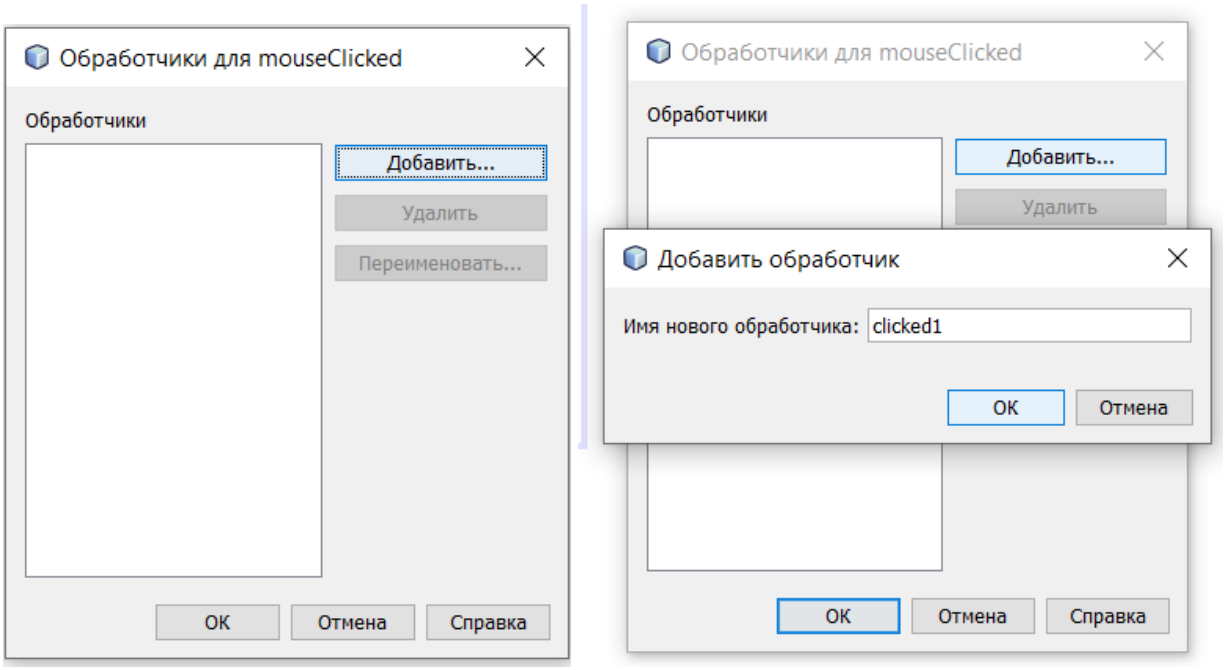


Рисунок 33 – Обработчик событий «mouseClicked»

После закрытия последнего окна NetBeans автоматически переведет вас в раздел Источник, где генерируется весь код по созданию формы.

```
private void clicked1(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
}
```

Рисунок 34 – Сгенерированный код обработчика событий «mouseClicked»

9. Теперь, чтобы понять с какой копкой работать с точки зрения ее имени в коде, перейдите во вкладку Проект - нажмите ПКМ по кнопке (в данном случае кнопка с текстом 1) – выберите пункт Изменить имя переменной… Если после jButton стоит цифра не соответствующая тексту у самой кнопки, то можно изменить, либо оставить как есть, т.к. это не влияет на саму работу формы.

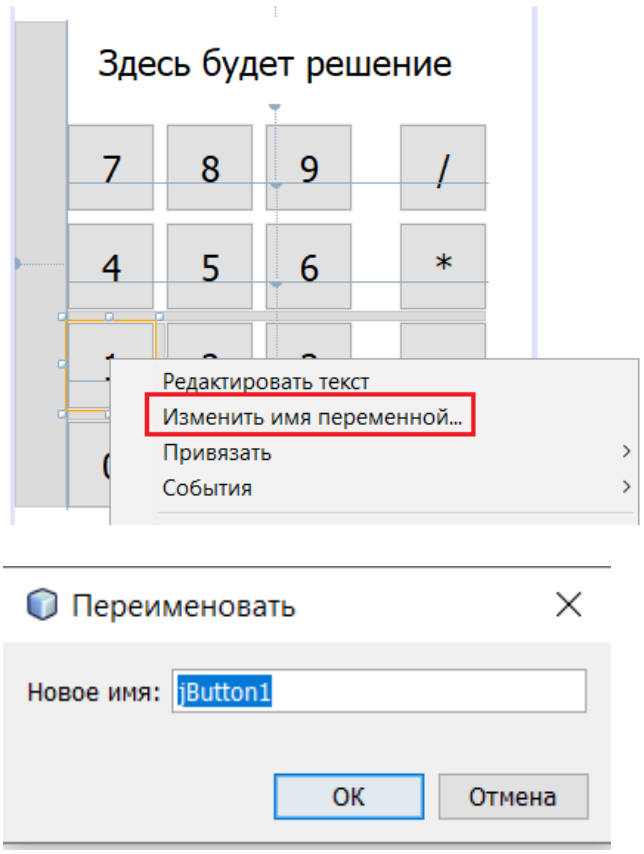


Рисунок 35 – Изменение имени переменной

10. Снова зайдите в раздел Источник, далее в метод, созданный для обработки нажатия кнопки. Заполните метод кодом, показанным на рисунке ниже.

```
private void clicked1(java.awt.event.MouseEvent evt) {
    jb1 = Integer.parseInt(jButton1.getText());
    jLabel1.setText(jLabel1.getText() + String.valueOf(jb1));
}
```

Рисунок 36 – Листинг обработчика событий «mouseClicked»

Чтобы получить цифру кнопки при нажатии пользователем на нее, то нужен метод getText(), но это текст, а не цифра. Чтобы преобразовать числовой текст в число, причем целое, нужен метод Integer.parseInt(). Для отображения текста в JLabel используется метод setText(), но так как jb1 является числом, то следует конвертировать его в текст при помощи метода String.ValueOf().

11. Для того, чтобы увидеть наше окно в действии, выберите класс Calculator и методе main вставьте следующий код:

```
public static void main(String[] args) {
    CalculatorFrame cf = new CalculatorFrame(); //Создан экземпляр класса
    cf.setVisible(true);    //Делает окно видимым (true)
}
```

Рисунок 37 – Листинг класса Calculator

12. Для оставшихся кнопок с цифрами (0, 2-9) повторите пункты 8, 9 и 10.

13. В разделе Источник в самом низу кода создайте код:

```
int jb1, jb2, jb3, jb4, jb5, jb6, jb7, jb8, jb9, jb0, result;
String jbZnak, label;
// Variables declaration - do not modify
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton10;
private javax.swing.JButton jButton11;
private javax.swing.JButton jButton12;
```

Рисунок 38 – Листинг переменных класса CalculatorFrame

14. Создайте слушатели на кнопки +, -, *, /.

```
private void clickedPlus(java.awt.event.MouseEvent evt) {
    label = jLabel1.getText();
    jLabel1.setText("");
    jbZnak = jButton17.getText();
}

private void clickedMinus(java.awt.event.MouseEvent evt) {
    label = jLabel1.getText();
    jLabel1.setText("");
    jbZnak = jButton16.getText();
}

private void clickedUmnoj(java.awt.event.MouseEvent evt) {
    label = jLabel1.getText();
    jLabel1.setText("");
    jbZnak = jButton15.getText();
}

private void clickedDel(java.awt.event.MouseEvent evt) {
    label = jLabel1.getText();
    jLabel1.setText("");
    jbZnak = jButton14.getText();
}
```

Рисунок 39 – Листинг обработчиков событий кнопок +, -, * и /

15. Для кнопки равно «=» и очистить «C» сделайте слушатель и напишите следующий код:

```
private void clickedRavno(java.awt.event.MouseEvent evt) {
    String label1 = jLabel1.getText();

    if(jbZnak.equals("+")){
        result = Integer.parseInt(label) + Integer.parseInt(label1);
        jLabel1.setText(String.valueOf(result));
```

Рисунок 40 – Листинг обработчика событий кнопок =

В методе clickedClear заменяется исходный текст JLabel на пустую строку.

В методе clickedRavno извлекается текст при помощи getText(), далее ищется индекс символа +. В операторе условия if проверяется сравнение

символа по найденному индексу с символом +. Если совпадает, то происходит извлечение первой подстроки number1 из строки label, начиная с первого символа с нулевым индексом по индекс найденного символа +. Извлечение второй подстроки number2 из строки label, начинается с индекса символа + по последний индекс строки label.

16. Проверьте правильность работы сложения с числами различной длины.

17. Сделайте тоже самое для вычитания, умножения и деления.

САМОСТОЯТЕЛЬНОЕ ЗАДАНИЕ

1. Сделать кнопку возведения числа в степень.
2. Сделать кнопку для извлечения корня n степени. $\sqrt[n]{a} = a^{\frac{1}{n}}$
3. Изменить внешний вид кнопок.

Лабораторная работа № 9

«Графика в Java»

Цель: научиться реализовывать векторные изображения при помощи библиотеки Graphics.

Общие компетенции:

ОК 1. Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.

ОК 4. Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.

ОК 5. Использовать информационно-коммуникационные технологии в профессиональной деятельности.

ОК 8. Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации.

Профессиональные компетенции:

ПК 2.2. Разрабатывать и публиковать программное обеспечение и информационные ресурсы отраслевой направленности со статическим и динамическим контентом на основе готовых спецификаций и стандартов.

ПК 2.3. Проводить отладку и тестирование программного обеспечения отраслевой направленности.

ПК 2.4. Проводить адаптацию отраслевого программного обеспечения.

ПК 2.5. Разрабатывать и вести проектную и техническую документацию.

ПК 2.6. Участвовать в измерении и контроле качества продуктов.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Как начертить прямую линию?

```
g.drawLine(20, 30, 360, 30);
```

здесь 20, 30 — это координаты x, y начала линии,

360, 30 — координаты конца линии.

Как задать цвет?

Метод setColor класса Graphics сделает текущим новый цвет:

```
// Запоминаем исходный цвет;
Color oldColor = g.getColor();
// Создаём синий цвет;
Color newColor = new Color(0, 0, 255);
```

```
// Устанавливаем синий цвет;
g.setColor(newColor);
// Чертим линию синим цветом;
g.drawLine(20, 30, 360, 30);
// Восстанавливаем исходный цвет;
g.setColor(oldColor);
```

Аргументы конструктора new Color(0, 0, 255) — это красный, зелёный и синий цвета соответственно (rgb).

Как задать rgb цвета?

В примере задан чисто синий цвет, т.к. значения других составляющих равны нулю. Вот чисто красный цвет:

```
Color newColor = new Color(255, 0, 0);
```

А это чисто зеленый цвет:

```
Color newColor = new Color(0, 255, 0);
```

Значения составляющих цвета изменяются от 0 до 255.

Светло-синий цвет, который мы использовали для заливки прямоугольника:

```
newColor = new Color(0, 215, 255);
```

Как задать цвет фона?

Задать цвет фона можно методом setBackground:

```
mainFrame.setBackground(Color.white);
```

Как нарисовать прямоугольник?

Методом drawRect класса Graphics:

```
g.drawRect(20, 40, 340, 20);
```

20, 40 — это координаты верхнего левого угла прямоугольника;

340 — длина;

20 — высота прямоугольника.

Как залить прямоугольник цветом?

Методом fillRect класса Graphics:

```
newColor = new Color(0, 215, 255);
g.setColor(newColor);
g.fillRect(21, 41, 339, 19);
g.setColor(oldColor);
```

Как нарисовать прямоугольник с закругленными углами?

Методом drawRoundRect класса Graphics.

Сопряжение, т.е. закругление на углах, делается с помощью частей овала.

```
g.drawRoundRect(20, 70, 340, 30, 20, 15);
```

первые 4 аргумента как у обычного прямоугольника. Пятый аргумент — 20 — это ширина прямоугольника, в который вписана часть овала сопряжения. Шестой аргумент — 15 — это высота прямоугольника, в который вписана часть овала сопряжения.

Как нарисовать овал?

Методом drawOval класса Graphics:

```
g.drawOval(20, 110, 150, 60);
```

Аргументы определяют прямоугольник, в который вписан овал.

Как нарисовать окружность?

Методом drawOval класса Graphics:

```
g.drawOval(200, 110, 60, 60);
```

Аргументы определяют прямоугольник, в который вписана окружность. Здесь рисуем овал, но длина и высота описанного прямоугольника равны, что и даёт окружность.

Как нарисовать дугу?

Методом drawArc класса Graphics:

```
g.drawArc(280, 110, 80, 60, 0, 180);
```

первые 4 аргумента как у обычного прямоугольника. Пятый аргумент — 0 — это угол, от которого отсчитывается угол самой дуги. 180 — это угол дуги. Углы отсчитывают от горизонтальной оси: по часовой стрелке отрицательное направление, протв — положительное. В примере 180 градусов (величина дуги) отсчитываем от горизонтальной линии.

Как нарисовать многоугольник?

Методом drawPolygon класса Graphics:

```
int[] arrayX = {20, 100, 100, 250, 250, 20, 20, 50};
int[] arrayY = {180, 180, 200, 200, 220, 200, 200, 190};
Polygon poly = new Polygon(arrayX, arrayY, 8);
g.drawPolygon(poly);
```

Здесь создаём объект класса Polygon. arrayX — это x-координаты вершин многоугольника, arrayY — это y-координаты вершин многоугольника, 8 — число вершин многоугольника.

Как создать объект точки?

Для этого используем класс Point:

```
Point aPoint = new Point(50, 190);
```

аргументы — это x, y координаты.

Как определить, что точка принадлежит многоугольнику?

```
Polygon poly = new Polygon(arrayX, arrayY, 8);
g.drawPolygon(poly);
Point aPoint = new Point(50, 190);
if(poly.contains(aPoint))
{
   g.drawString(«Yes», 50, 190);
}
```

Используем метод класса Polygon contains для определения лежит ли точка в многоугольнике.

Как вывести строку?

Методом drawString класса Graphics:

```
g.drawString(«Yes», 50, 190);
```

строка «Yes» будет выведена от точки с координатами 50, 190.

Как задать шрифт?

Для этого используем класс Font:

```
Font font = new Font(«Tahoma», Font.BOLD|Font.ITALIC, 40);
```

где «Tahoma» — название шрифта,

Font.BOLD|Font.ITALIC — жирный шрифт с наклоном,

40 — высота шрифта.

После задания шрифта мы делаем его текущим и выводим строку этим шрифтом:

```
g.setFont(font);
g.drawString(«SBP», 270, 220);
```

Как задать цвет текста?

Чтоб задать цвет текста создадим и установим в графический контекст новый цвет:

```
newColor = new Color(0, 0, 255);
g.setColor(newColor);
```

Здесь мы создали чисто синий цвет. А теперь выводим строку синим цветом:

```
g.drawString(«SBP», 270, 220);
```

ПРАКТИЧЕСКАЯ ЧАСТЬ

1. Создайте новый проект и назовите его Window

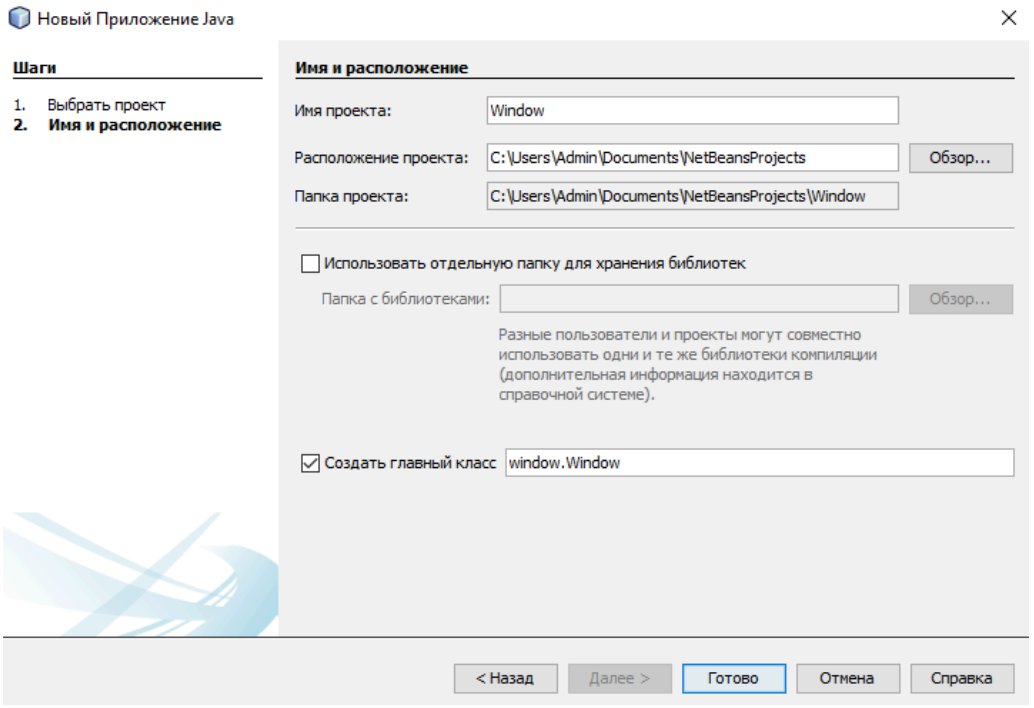


Рисунок 41 – Создание проекта Window

2. Класс Window будет запускаться при запуске приложения, прорисовывать окно и элементы на нём.

В классе Window напишите следующий код:

```
import java.awt.BorderLayout;
import javax.swing.JFrame;

public class Window {

    public static void main(String[] args) {
        /* Задание заголовка окна*/
        JFrame w=new JFrame("Окно с изображением");
        /*Задание размеров окна*/
        w.setSize(400, 400);

        /*Если у окна не будет функции закрытия,
          при нажатии крестика окно не закроется.*/
        w.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        /*Менеджер определяет
          каким образом в окне расположены объекты.*/
        w.setLayout(new BorderLayout(1,1));
    }

}
```

Рисунок 42 – Листинг класса Window

Добавьте необходимые операторы импорта.

3. В пакете вашего проекта window создайте еще один класс и назовите его Canvas. Этот класс будет задавать настройки видимости и прорисовывать все графические объекты.

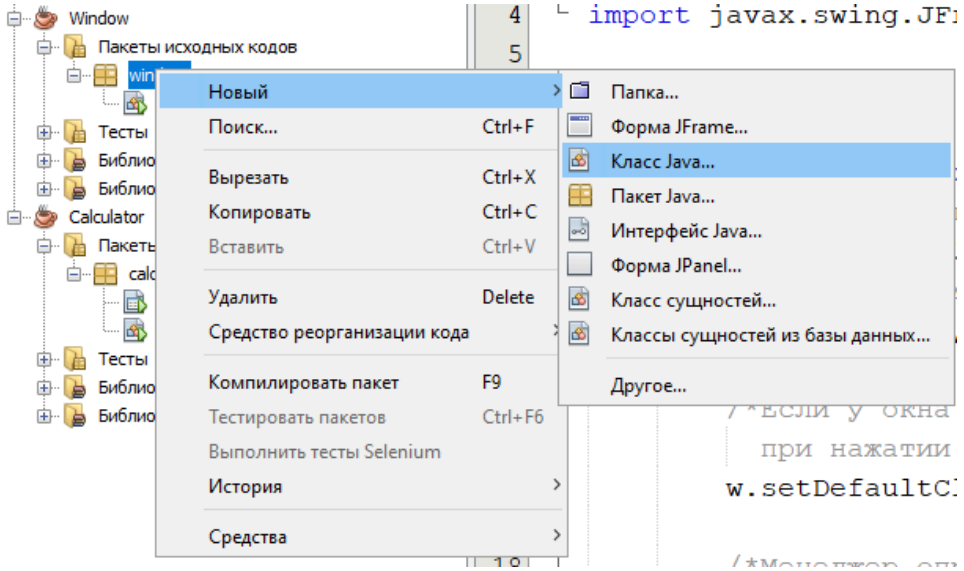


Рисунок 43 – Меню создания нового класса

4. В классе Canvas напишите следующий код:

```
/* Класс, который будет рисовать элементы*/
public class Canvas extends JComponent{

/*Метод, перерисовывающий элемент внутри окна
 *при обновлении*/
  public void paintComponent(Graphics g){
        super.paintComponents(g);
        Graphics2D g2d=(Graphics2D)g;

        g2d.setPaint(Color.GREEN); //Устанавливает цвет рисования в зелёный
        g2d.drawRect(100, 100, 80, 20); //Рисует текущим цветом прямоугольник
        g2d.setPaint(Color.RED); //Устанавливает цвет рисования в красный
        //Рисует текущим цветом в координатах (150,150) строку "привет мир"
        g2d.drawString("Привет мир", 150, 150);
        g2d.setColor(Color.blue); //Устанавливает цвет рисования голубой
        g2d.fillOval(200, 50, 50, 20); //Рисует текущим цветом овал в координатах (200,50)

        super.repaint(); //Вызывает обновление себя после завершения рисования
   }
}
```

Рисунок 44 – Листинг класса Canvas

5. Вернитесь в класс Window и после строки w.setLayout(new BorderLayout(1,1)); добавьте вызов класса Canvas:

```
/*Менеджер определяет
  каким образом в окне расположены объекты.*/
w.setLayout(new BorderLayout(1,1));

Canvas canv=new Canvas();
w.add(canv);
w.setVisible(true);
```

6. В результате при запуске проекта должно получится следующее:

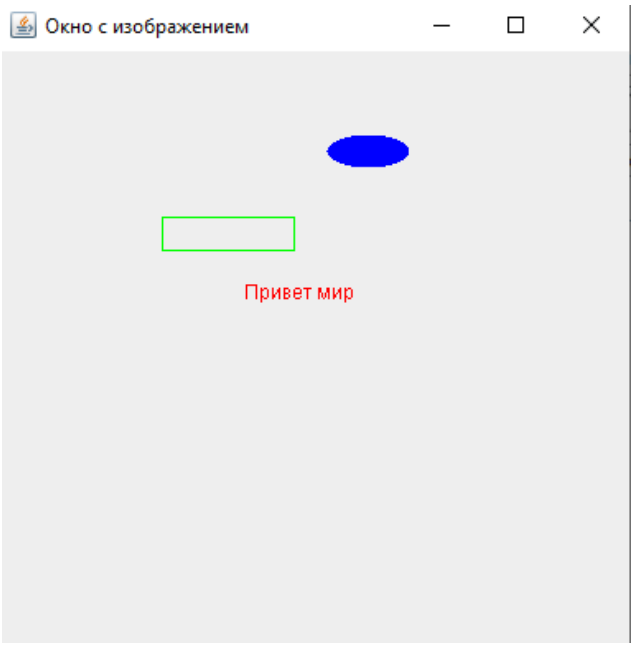
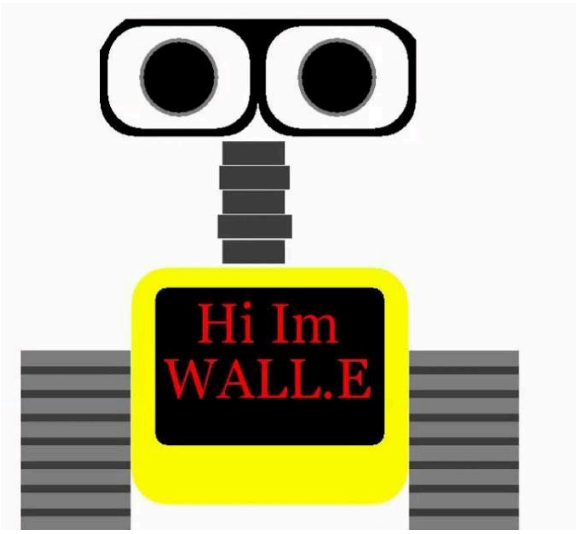
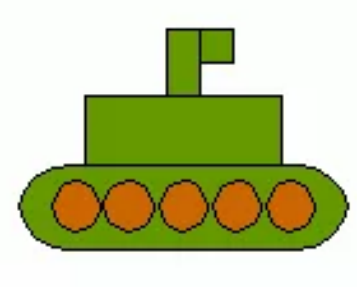


Рисунок 45 – Окно вывода проекта Window

САМОСТОЯТЕЛЬНЫЕ ЗАДАНИЕ

Задание: Создайте рисунок по образцу. Все детали и части рисунка описываются в классе Canvas.


Hi Im
WALL.E



Лабораторная работа № 10

«Установка Android Studio. Разработка программ для ОС Android»

Цель: научиться:

- разрабатывать Android-приложения;
- устанавливать Android-приложения на телефон;

создавать Android-приложения в Android Studio

Общие компетенции:

ОК 1. Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.

ОК 4. Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.

ОК 5. Использовать информационно-коммуникационные технологии в профессиональной деятельности.

ОК 8. Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации.

Профессиональные компетенции:

ПК 2.2. Разрабатывать и публиковать программное обеспечение и информационные ресурсы отраслевой направленности со статическим и динамическим контентом на основе готовых спецификаций и стандартов.

ПК 2.3. Проводить отладку и тестирование программного обеспечения отраслевой направленности.

ПК 2.4. Проводить адаптацию отраслевого программного обеспечения.

ПК 2.5. Разрабатывать и вести проектную и техническую документацию.

ПК 2.6. Участвовать в измерении и контроле качества продуктов.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

В первую очередь для создания приложений нужно загрузить и установить пакет JDK 8, который необходим для разработки на языке Java. JDK 8 можно найти на сайте компании Oracle. Ссылка на пакет JDK 8: http://www.oracle.com/technetwork/java/javase/downloads/index.html.

Существуют разные среды разработки для Android, но рекомендуемой средой разработки является Android Studio. Загрузить файл установщика можно с официального сайта: http://developer.android.com/sdk/index.html.

Для скачивания пакета установки для OS Windows надо нажать на кнопку "Download Android Studio":

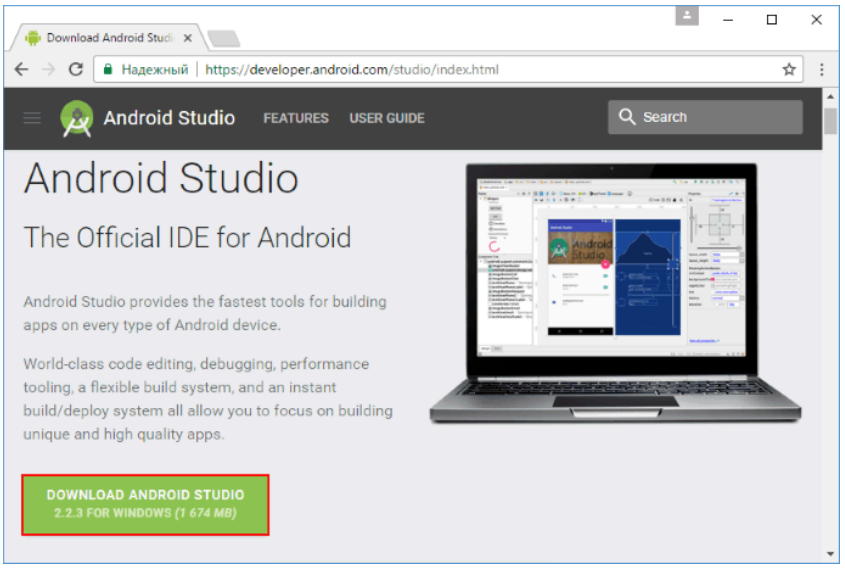


Рисунок 46 - Официальный сайт для скачивания Android Studio

Установка Android Studio

Ничего необычного в установке нет - обычный диалог инсталлятора. В процессе нужно будет ответить лишь на один важный вопрос, и то это опционально:

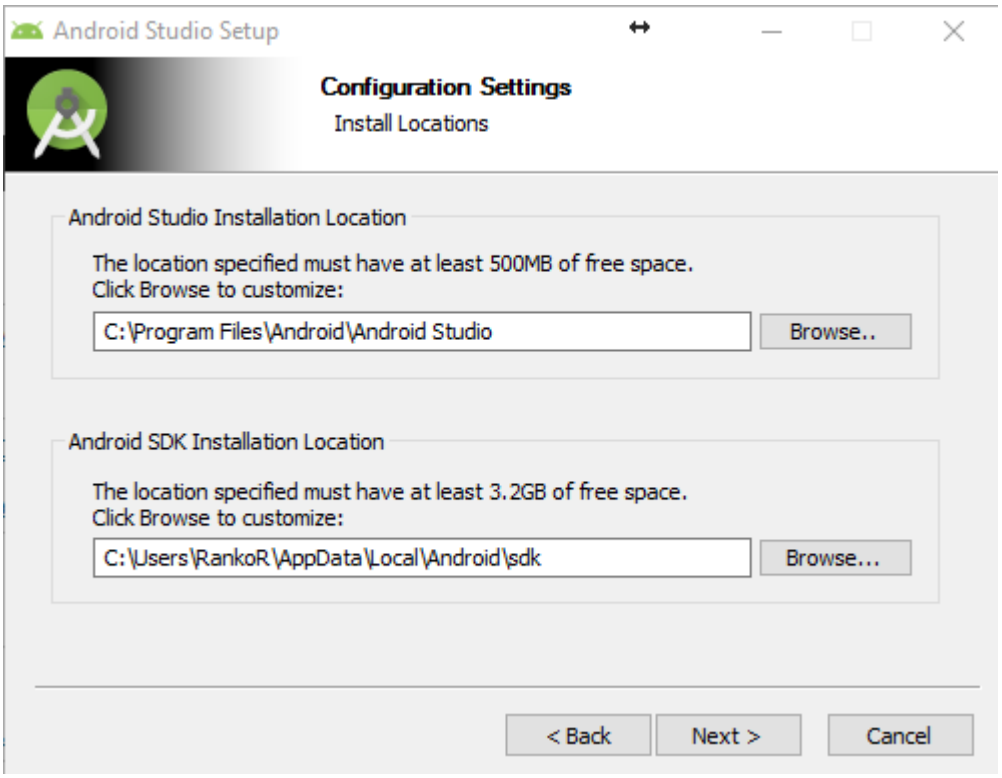


Рисунок 47 - Этап выбора пути установки

Как следует из скриншота (рисунок 4), установщик спрашивает, куда ставить студию, и куда ставить SDK. С SDK нужно быть внимательным.
Во-первых, для установки SDK нужно минимум 3.2 GB места на диске.
Во-вторых, путь установки SDK и Android Studio должен содержать английский алфавит.

После этого понадобится стандартно несколько раз нажать на кнопочку «далее», и на этом установка Android Studio завершена.

Запуск Android Studio

При первом запуске вам будет предложено установить в диалоговых окнах несколько параметров конфигурации. В первом диалоговом окне основное внимание уделяется импорту настроек из ранее установленной версии Android Studio:

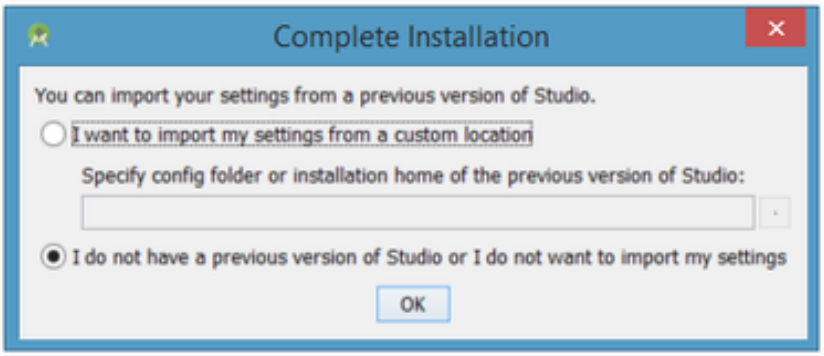


Рисунок 48 - Параметры импорта

Можно принять настройки по умолчанию и нажать на кнопку «ОК». После этого Android Studio выведет диалоговое окно «Мастера установки»:

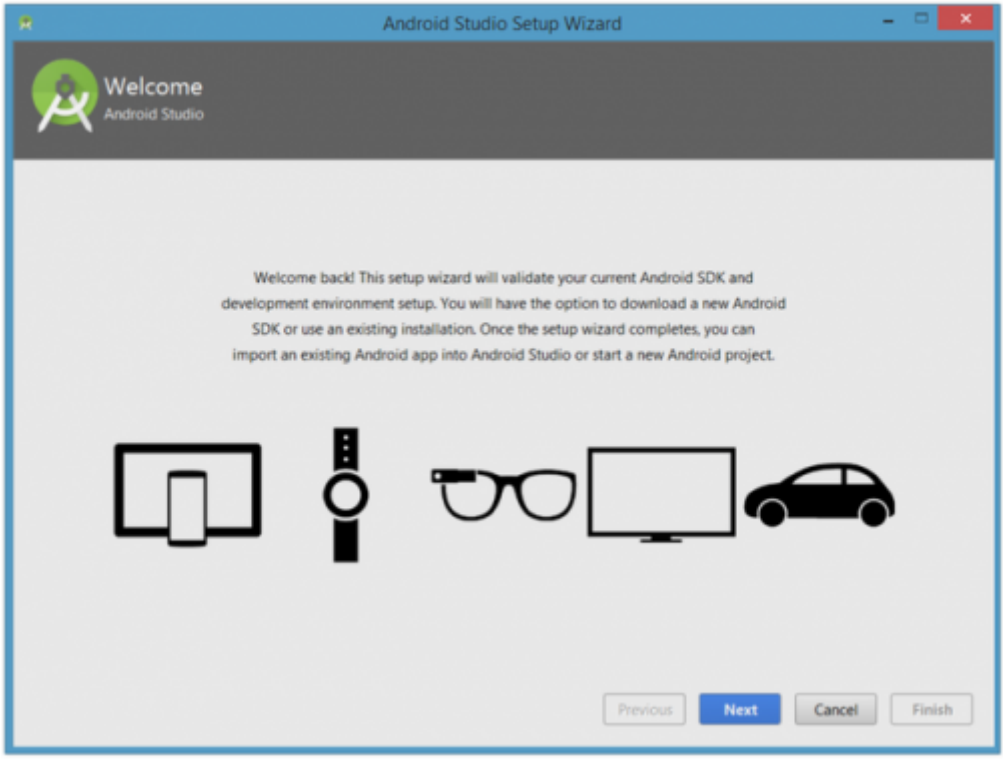


Рисунок 49 - Проверка настроек Android SDK и среды разработки

После нажатия кнопки «Далее», «Мастер установки» предложит выбрать тип установки компонентов SDK. На данный момент я рекомендую использовать стандартную конфигурацию:

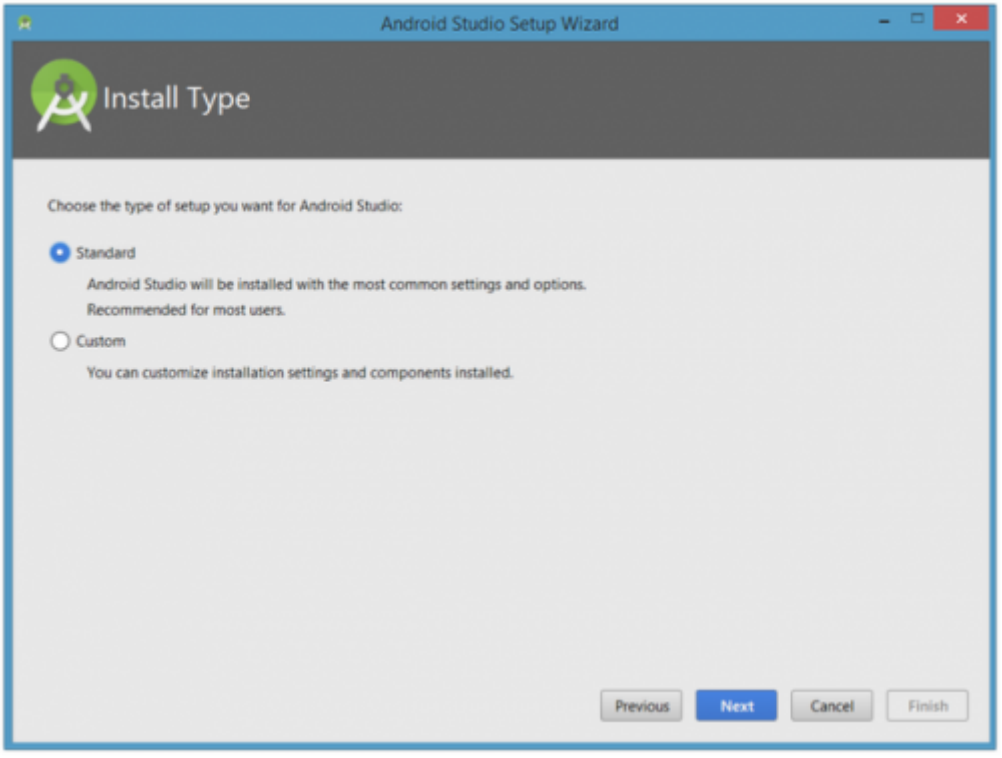


Рисунок 50 - Выбор типа установки

Нажмите кнопку «Далее» и подтвердите выбранные настройки. Затем нажмите кнопку «Готово», чтобы продолжить:

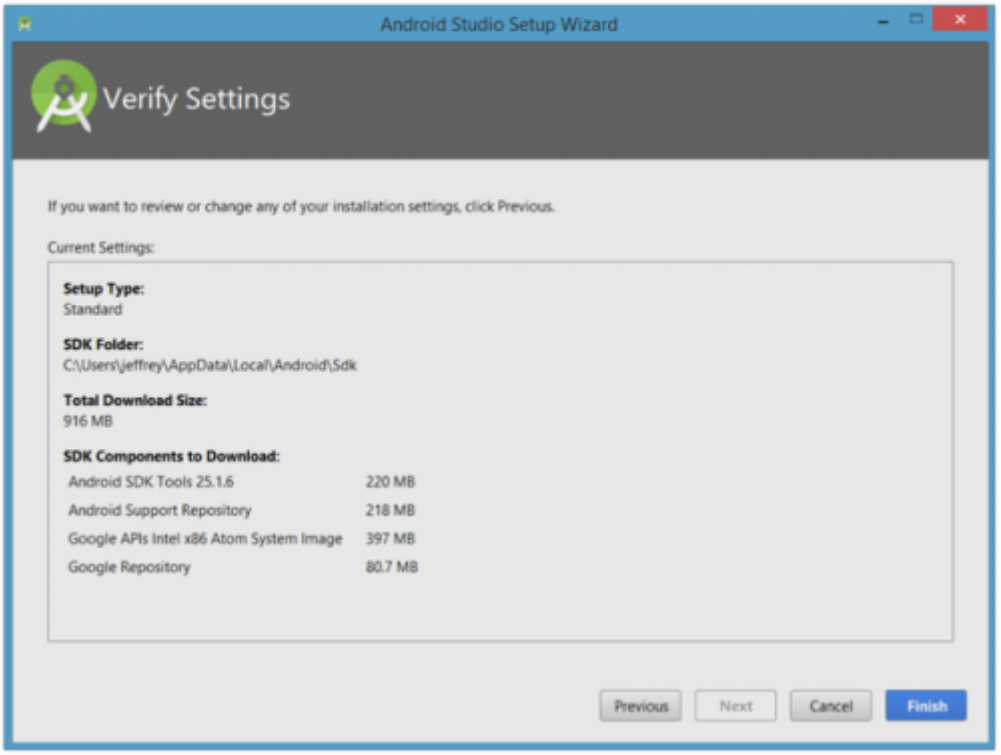


Рисунок 51 - Просмотр настроек

Нажмите кнопку «Готово», чтобы завершить работу «Мастера установки». После этого вы увидите диалоговое окно «Добро пожаловать в Android Studio»:

Рисунок 52 - Добро пожаловать в Android Studio

Оно используется для запуска нового проекта Android Studio (start a new Android Studio project), работы с существующим проектом (Open an existing Android Studio project) и т. д.

ПРАКТИЧЕСКАЯ ЧАСТЬ

Android Studio и создание первого проекта

Откроейте Android Studio. На начальном экране выберем пункт Start new Android Project (рисунок 9).

После этого отобразится диалоговое окно создания нового проекта:

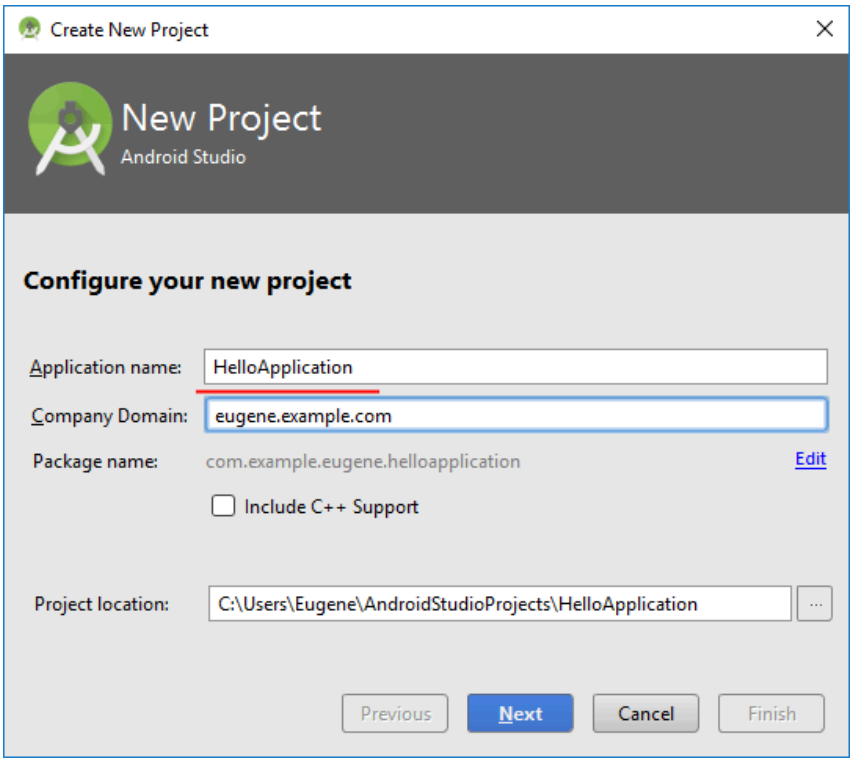


Рисунок 53 - Первый этап создания проекта

В окне создания нового проекта можно установить его начальные настройки:

- В поле Application Name вводится название приложения;
- В поле Company Domain указывается домен приложения или тот пакет классов, где будет размещаться главный класс приложения;
- В поле Project Location можно установить расположение файлов проекта на жестком диске.

Далее нажмите на кнопку Next. Следующий шаг:

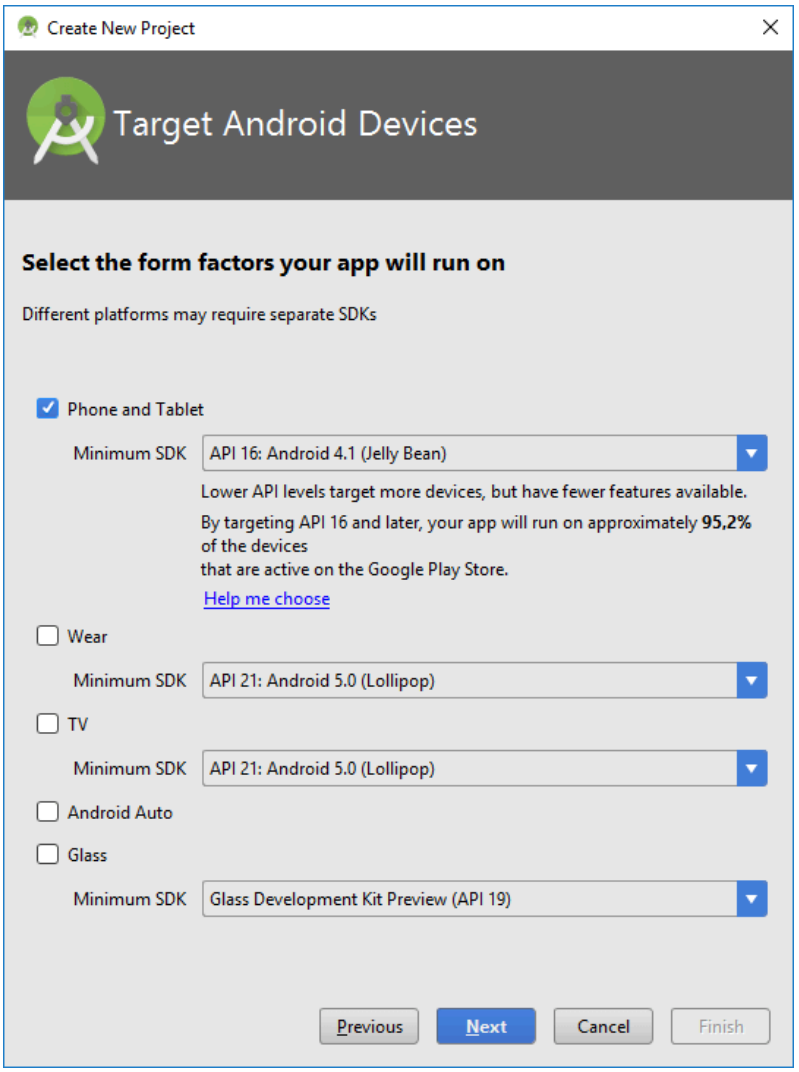


Рисунок 54 - Выбор минимальной поддерживаемой версии проекта

На этом шаге будет предложено установить минимальную поддерживаемую версию проекта. По умолчанию устанавливается версия Android 4.1, что покрывает более 95% устройств Android.

На следующем шаге надо выбрать шаблон проекта:

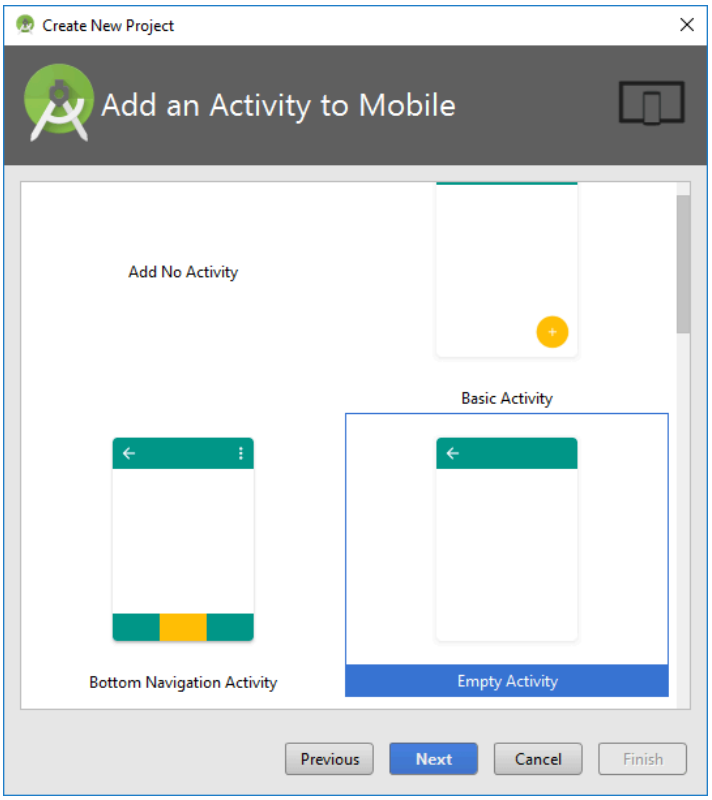


Рисунок 55 - шаблон проекта

Android Studio предоставляет ряд шаблонов для различных ситуаций, но самыми распространенными являются Basic Activity и Empty Activity. Это самые удобные шаблоны для старта для создания большинства приложений. Выберете шаблон Empty Activity.

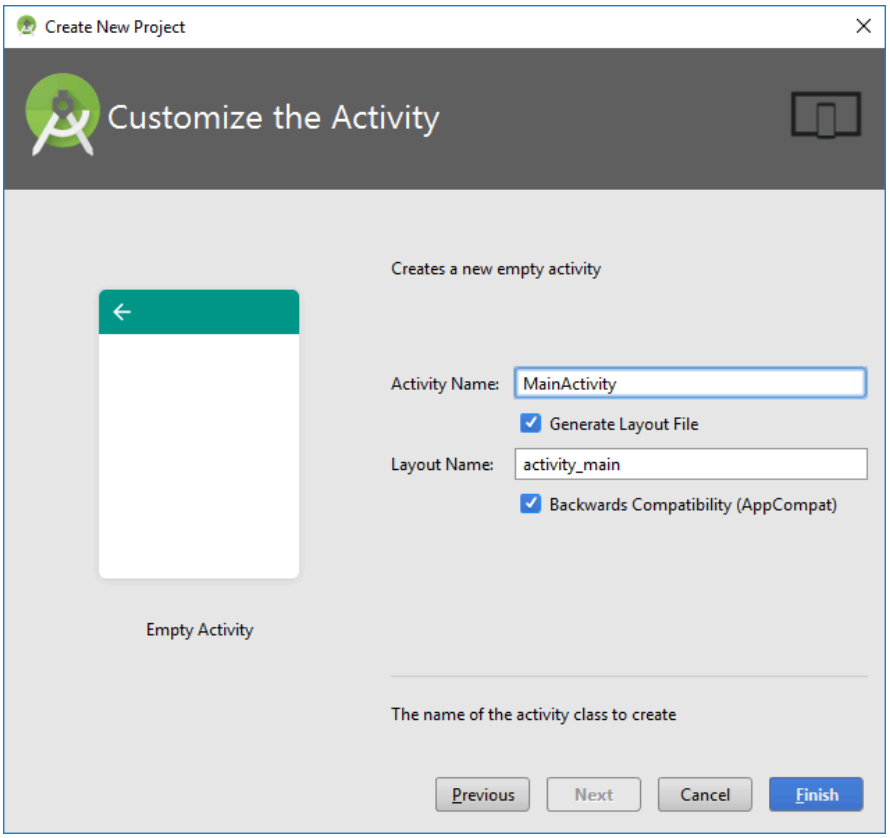


Рисунок 56 - Настройки проекта

При выборе Empty Activity на следующем шаге надо установить ряд настроек проекта:

- Activity Name: название главного класса приложения;
- Layout Name: название файла xml, в котором будет храниться определение визуального интерфейса;
- Generate Layout File: надо ли генерировать файл xml с определением визуального интерфейса.

Backwards Compatibility (AppCompat): в отмеченном состоянии позволяет установить обратную зависимость между различными версиями Android.

Нажав на кнопку Finish через некоторое время Android Studio создаст и откроет проект:

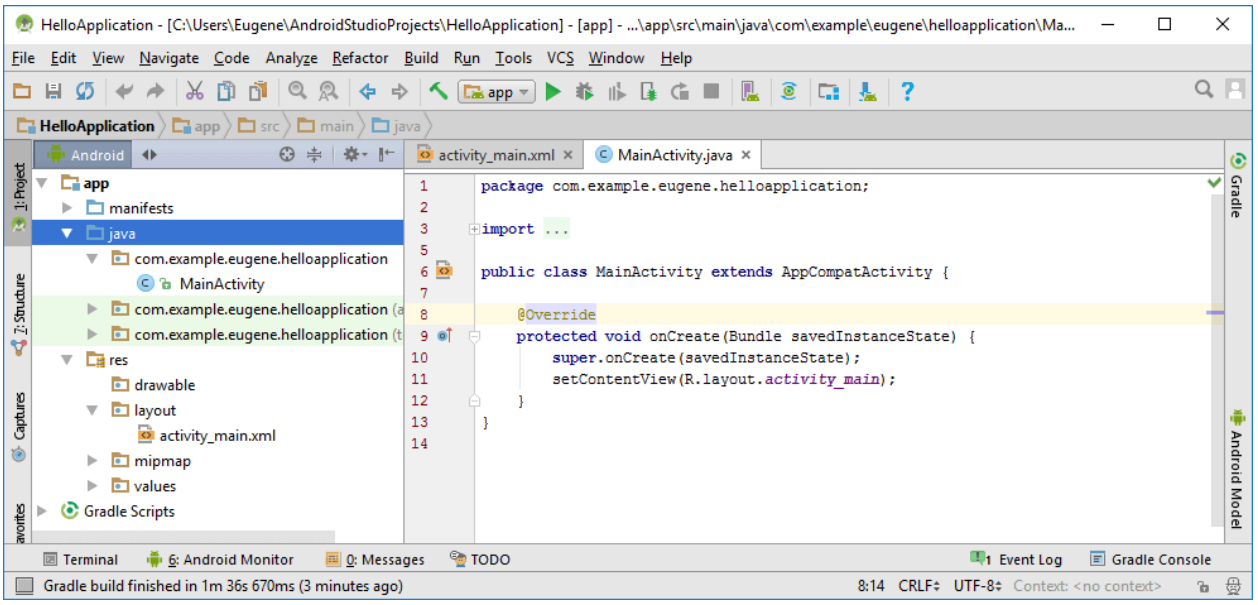


Рисунок 57 - Новый проект

Режим разработчика на телефоне.

Для запуска и тестирования приложения можно использовать эмуляторы или реальные устройства. Но в идеале лучше тестировать на реальных устройствах.

Для использования мобильного устройства для тестирования на рабочую машину необходимо установить драйвер. Если смартфон от Google - Nexus 5/6/5x/6P или Google Pixel, то для его поддержки необходимо через SDK Manager установить пакет Google Usb Driver. Если же производитель аппарата - другой вендор, то надо установить тот USB-драйвер, который поставляется данным вендором. Если ОС - Windows 10, то там, как правило, система сама может найти через центр обновлений драйвер и установить его.

По умолчанию опции разработчика на смартфонах скрыты. Чтобы сделать их доступными, надо зайти в Settings > About phone (Настройки > О телефоне) и семь раз нажать Build Number (Номер сборки) (рисунок 15).

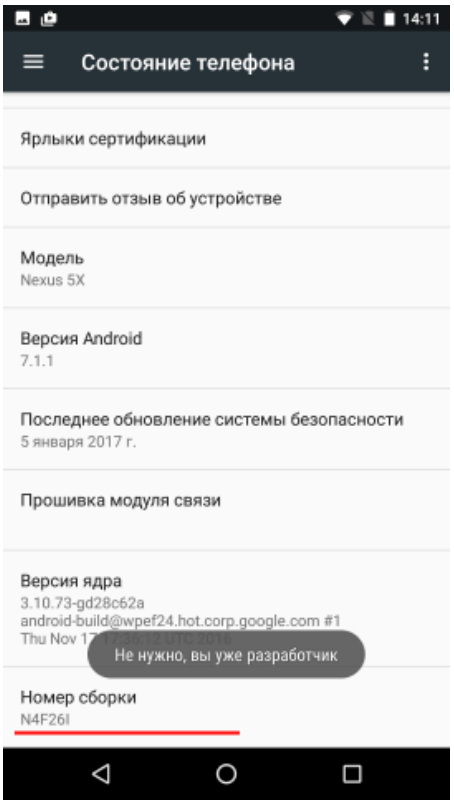


Рисунок 58 - Режим разработчика

Вернувшись назад вы увидите доступный пункт Developer options (Для разработчика). Перейдите к пункту Для разработчиков и включим возможность отладки по USB:

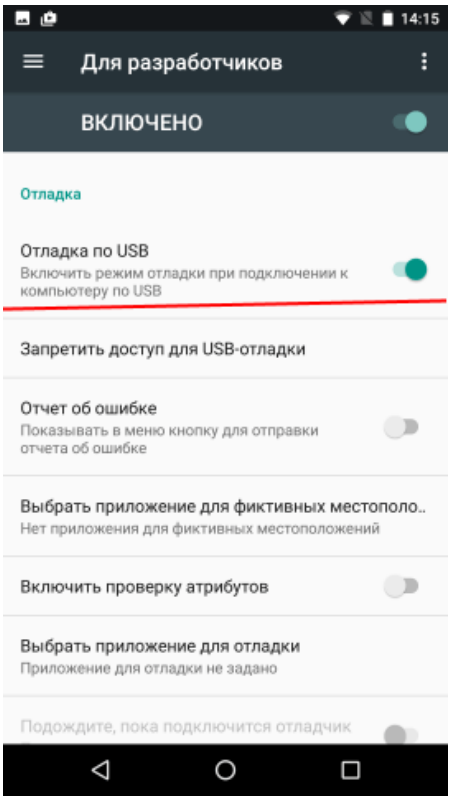


Рисунок 59 - Отладка по USB

Затем подключите устройство с ОС Android (если тестируете на реальном устройстве) и запустите проект, нажав на зеленую стрелочку на панели инструментов (рисунок 17).

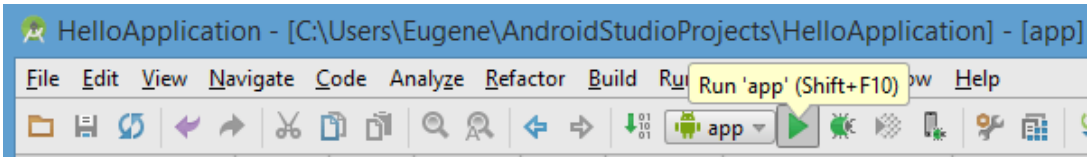


Рисунок 60 - Кнопка запуска проекта

Затем начнется построение проекта. Данный процесс может занять некоторое время, после чего отобразится диалоговое окно для выбора устройства для запуска. Здесь можно выбрать подключенный к компьютеру гаджет, либо эмулятор:

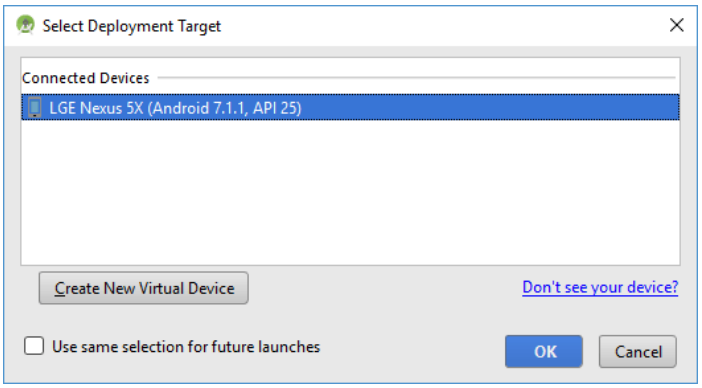


Рисунок 61 - Выбор устройства для запуска проекта

Выберем устройство и нажмите на кнопку OK. После запуска увидите приложение на экране устройства:

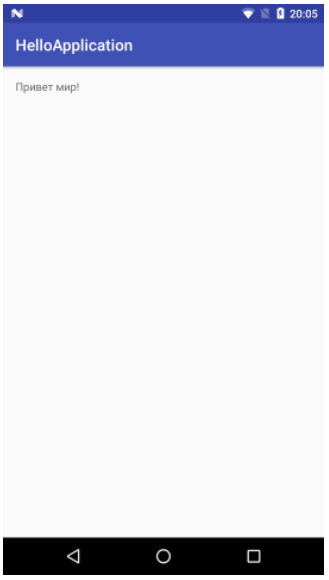


Рисунок 62 - Первое приложение на мобильном устройстве

Лабораторная работа № 11

«Разработка программ для ОС Android»

Цель: научиться:

- создавать приложения для ОС Android;
- обрабатывать нажатия на кнопки;
- получать введенный пользователем текст;
- выводить результат на экран мобильного приложения

Общие компетенции:

ОК 1. Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.

ОК 4. Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.

ОК 5. Использовать информационно-коммуникационные технологии в профессиональной деятельности.

ОК 8. Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации.

Профессиональные компетенции:

ПК 2.2. Разрабатывать и публиковать программное обеспечение и информационные ресурсы отраслевой направленности со статическим и динамическим контентом на основе готовых спецификаций и стандартов.

ПК 2.3. Проводить отладку и тестирование программного обеспечения отраслевой направленности.

ПК 2.4. Проводить адаптацию отраслевого программного обеспечения.

ПК 2.5. Разрабатывать и вести проектную и техническую документацию.

ПК 2.6. Участвовать в измерении и контроле качества продуктов.

ПРАКТИЧЕСКАЯ ЧАСТЬ

Зная некоторые основы компоновки и такие элементы как TextView, EditText и Button, уже можно составить более-менее полноценное приложение.

Создайте новый проект и определите в файле activity_main.xml следующий интерфейс:

Рисунок 63 - Простейший интерфейс калькулятора

Корневой контейнер компоновки должен представлять элемент LinearLayout с вертикальной ориентацией. Первый элемент в нем - горизонтальный элемент LinearLayout, в котором должны быть определены два текстовых поля TextView: одно для вывода результата вычислений и одно для вывода текущего знака операции.

Затем должен идти элемент EditText, предназначенный для ввода чисел.

И далее расположите четыре элемента LinearLayout с горизонтальными рядами кнопок. Чтобы все кнопки занимали равное пространство внутри контейнера, для них установите атрибуты android:layout_weight="1" и android:layout_width="0dp".

Кроме того, для числовых кнопок в качестве обработчика нажатия установите метод onNumberClick, а для кнопок со знаками операций атрибут onClick указывает на метод onOperationClick.

Теперь класс MainActivity должен содержать следующий код:

```
package com.example.eugene.calculator;

import android.support.v7.app.AppCompatActivity;
```

```
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
//текстовое поле для вывода результата
    TextView resultField;
//поле для ввода числа
    EditText numberField;
//текстовое поле для вывода знака операции
    TextView operationField;
// операнд операции
    Double operand = null;
// последняя операция
    String lastOperation = "=";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
   //метод, получающий все поля по id из activity_main.xml
       setContentView(R.layout.activity_main);
        resultField =(TextView) findViewById(R.id.resultField);
        numberField = (EditText) findViewById(R.id.numberField);
        operationField                =                (TextView)
findViewById(R.id.operationField);
    }
    // сохранение состояния
    @Override
   protected void onSaveInstanceState(Bundle outState) {
        outState.putString("OPERATION", lastOperation);
        if(operand!=null)
            outState.putDouble("OPERAND", operand);
        super.onSaveInstanceState(outState);
    }
    // получение ранее сохраненного состояния
    @Override
    protected         void         onRestoreInstanceState(Bundle
savedInstanceState) {
        super.onRestoreInstanceState(savedInstanceState);
        lastOperation                                          =
savedInstanceState.getString("OPERATION");
        operand= savedInstanceState.getDouble("OPERAND");
        resultField.setText(operand.toString());
        operationField.setText(lastOperation);
    }
    // обработка нажатия на числовую кнопку
    public void onNumberClick(View view){
        Button button = (Button)view;
        numberField.append(button.getText());
        if(lastOperation.equals("=") && operand!=null){
            operand = null;
```

```
        }
    }
    // обработка нажатия на кнопку операции
    public void onOperationClick(View view){
       Button button = (Button)view;
       String op = button.getText().toString();
         String number = numberField.getText().toString();
       // если введено что-нибудь
       if(number.length()>0){
            number = number.replace(',', '.');
            try{
          performOperation(Double.valueOf(number), op);
            }catch (NumberFormatException ex){
                numberField.setText("");
            }
        }
        lastOperation = op;
        operationField.setText(lastOperation);
    }

    private void performOperation(Double number, String
operation){
// если операнд ранее не был установлен (при вводе самой первой
операции)
        if(operand ==null){
            operand = number;
        }
        else{
            if(lastOperation.equals("=")){
                lastOperation = operation;
            }
            switch(lastOperation){
                case "=":
                    operand =number;
                    break;
                case "/":
                    if(number==0){
                        operand =0.0;
                    }
                    else{
                        operand /=number;
                    }
                    break;
                case "*":
                    operand *=number;
                    break;
                case "+":
                    operand +=number;
                    break;
                case "-":
                    operand -=number;
                    break;
            }
```

```
        }
      resultField.setText(operand.toString().replace('.', ','));
         numberField.setText("");
     }
}
```

Подробнее об этом классе. Вначале в методе onCreate() происходит получение всех полей из activity_main.xml, текст которых будет изменяться:3

```
resultField =(TextView) findViewById(R.id.resultField);
numberField = (EditText) findViewById(R.id.numberField);
operationField = (TextView) findViewById(R.id.operationField);
```

Результат операции будет попадать в переменную operand, которая представляет тип Double, а знак операции - в переменную lastOperation:

```
Double operand = null;
String lastOperation = "=";
```

Так как при переходе от портретной ориентации к альбомной или наоборот можно потерять все введенные данные, чтобы их не потерять, их нужно сохранить в методе onSaveInstanceState() и обратно получить в методе onRestoreInstanceState().

При нажатии на числовую кнопку будет вызываться метод onNumberClick, в котором добавляется введенная цифра или знак запятой к тексту в поле numberField:

```
Button button = (Button)view;
numberField.append(button.getText());
 if(lastOperation.equals("=") && operand!=null){
    operand = null;
}
```

При этом если последняя операция представляла собой получение результата (знак "равно"), тогда сбрасывается переменная operand.

В методе onOperationClick происходит обработка нажатия на кнопку со знаком операции:

```
Button button = (Button)view;
String op = button.getText().toString();
String number = numberField.getText().toString();
if(number.length()>0){
    number = number.replace(',', '.');
    try{
      performOperation(Double.valueOf(number), op);
    }catch (NumberFormatException ex){
        numberField.setText("");
    }
```

```
}
lastOperation = op;
operationField.setText(lastOperation);
```

Здесь происходит получение ранее введенного числа и введенную операцию и передача их в метод performOperation(). Так как в метод передается не просто строка, а число Double, то нужно преобразовать строку в число. И поскольку теоретически могут быть введены нечисловые символы, то для отлова исключения, которое может возникнуть при преобразовании используется конструкция try...catch.

Кроме того, так как разделителем целой и дробной части в Double в java является точка, то надо заменить запятую на точку, так как предполагается, что используется в качестве разделителя запятая.

А методе performOperation() выполняется собственно операция. При вводе первой операции, когда операнд еще не установлен, просто устанавливается операнд:

```
if(operand ==null){
    operand = number;
}
```

При вводе второй и последующих операций применяется предыдущая операция, знак которой хранится в переменной lastOperation, к операнду operand и второму числу, которое было введено в числовое поле. Полученный результат операции сохраняется в переменной operand.

САМОСТОЯТЕЛЬНАЯ ЧАСТЬ

1. Задайте фоновый цвет кнопок.
2. Увеличьте размер текста на кнопках и в поле EditText.

Добавьте кнопки и операции возведения в квадрат и в любую другую степень.