Лабораторна робота №1
Тема: «Знайомство з базовим синтаксисом та операторами Java»

МЕТА ЛАБОРАТОРНОЇ РОБОТИ:

• Встановити необхідне програмне забезпечення для програмування на ЯП Java;
• ознайомитися з базовим синтаксисом мови та основними операторами;
• вивчити приклади угоди щодо коду для ЯП Java.

ХОД РАБОТИ:

Для роботи з Java вам потрібно встановити JDK і середовище розробки.

Java Development Kit (скорочено JDK) - безкоштовно розповсюджуваний комплект розробника додатків на мові Java, що включає компілятор Java (javac), стандартні бібліотеки класів Java, приклади, документацію, різні утиліти і виконавчу систему Java (JRE).

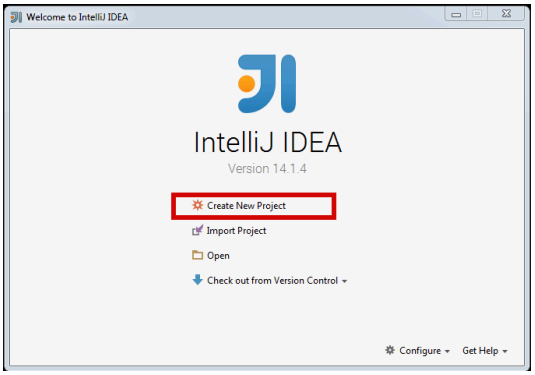
Завантажити JDK можна за посиланням - http://goo.gl/X68FzJ або можна загуглити JDK і вибрати перший результат пошуку. На сторінці виберіть розрядність та вашу ОС.

Як середовище розробки можете використовувати будь-яке зручне середовище. Для мови Java існує "велика трійка" середовищ розробки: NetBeans, IntelliJ IDEA та Eclipse. Особистою перевагою лектора є IDEA, NetBeans вважається більш доброзичливим для новачка.

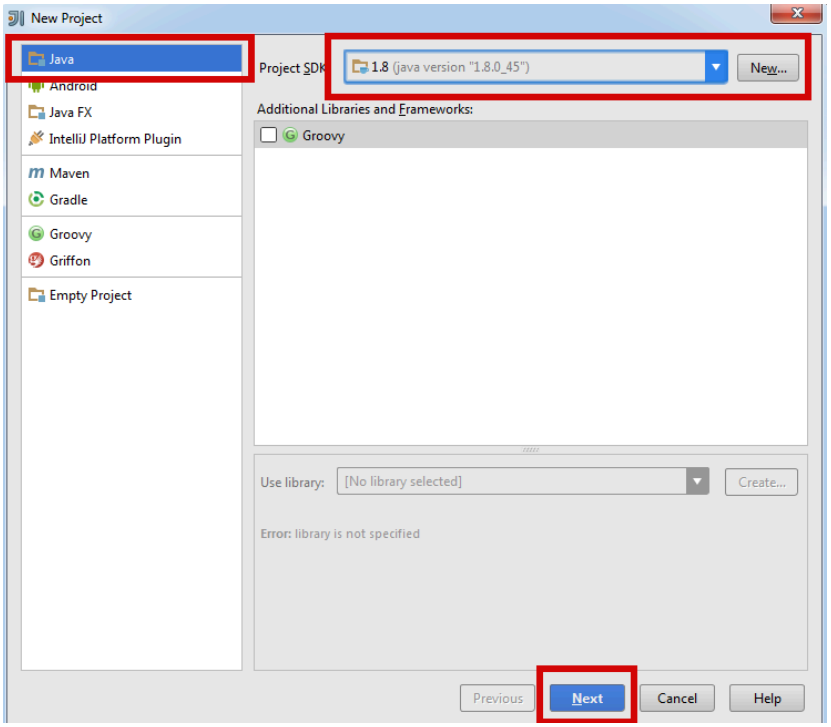
Щоб скачати IntelliJ IDEA, пройдіть за посиланням https://goo.gl/yBR7ux. Ви можете завантажити Ultimate версію, яка активна протягом 30 днів або безкоштовну версію Community. Поки вам достатньо Community версії, якщо для якоїсь лабораторної роботи потрібна буде Ultimate-версія, вам буде повідомлено заздалегідь.

1. Створення java-проекту в IntelliJ IDEA

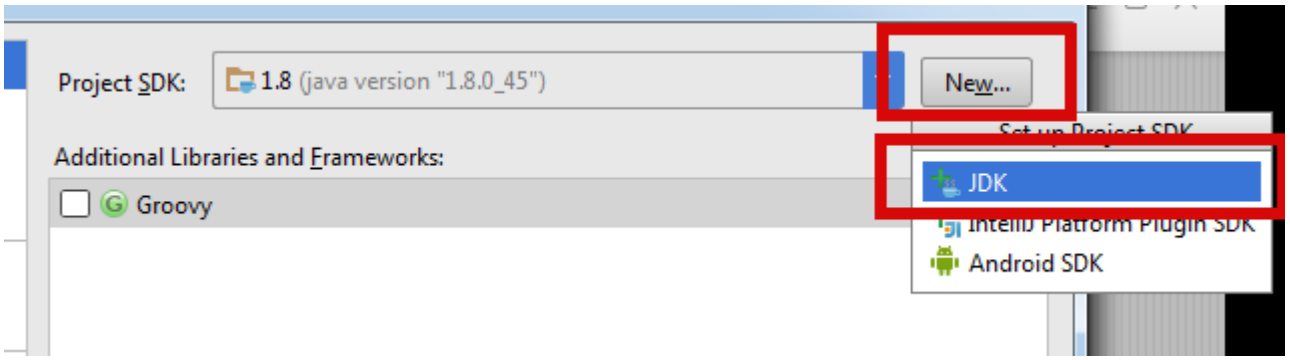
Запускаємо IDEA і вибираємо Create New Project



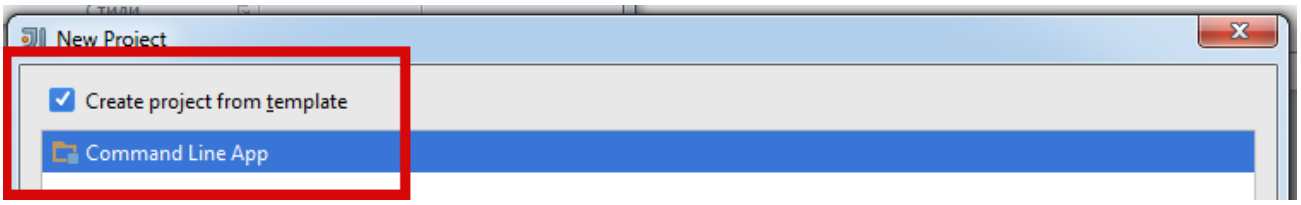
Далі вибираємо зліва Java, зверху вказуємо версію JDK і натискаємо Next.



Якщо IDEA не знає шлях і зверху порожнє поле, натисніть кнопку New… і вкажіть шлях до JDK (як правило, він встановлений у Program Files\Java\jdkXXX).

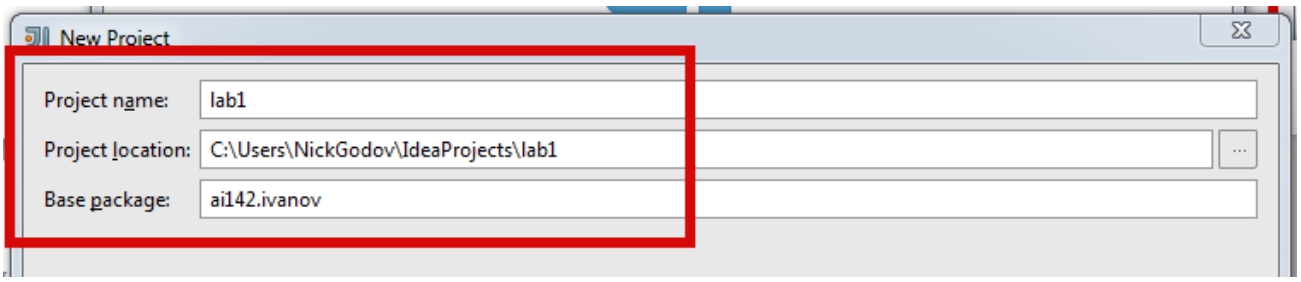


Виберемо шаблон Command Line App та натиснемо Next

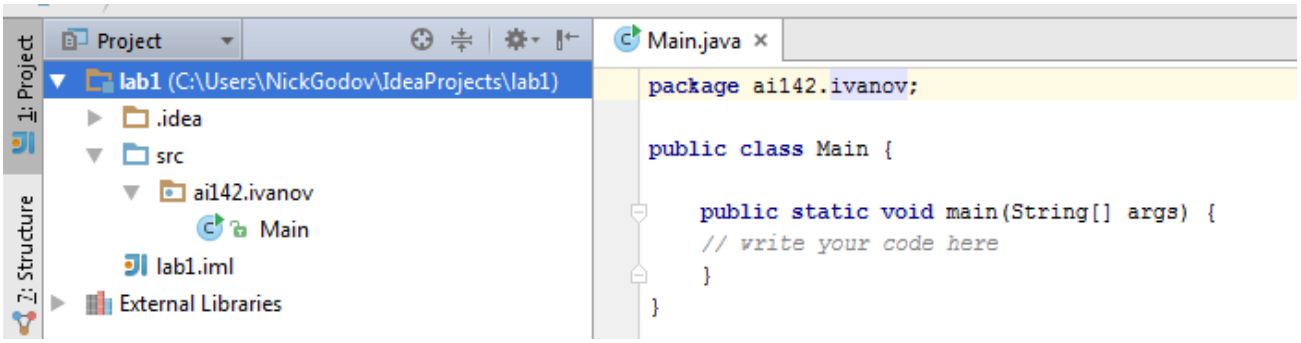


Далі вам потрібно вказати назву проекту, шлях до нього, а також Base package. Свого часу вам пояснює, що таке package і навіщо вони потрібні (хоча ви можете

загуглити це і самостійно), на даний момент від вас потрібно написати його так: [назва_групи].[прізвище]. Наприклад, ai141.ivanov або ai143.petrov



Далі, тиснемо Finish та IDEA створить проект. Одна з наступних лабораторних робіт, можливо, буде присвячена найближчому знайомству із середовищем розробки. На даний момент нам потрібен тільки файл Main.java, всередині якого ми і будемо працювати в цій лабораторній роботі.



Ознайомлення з основами мови Java

При розробці мови Java було взято за основу синтаксис мов C і C++, тому деякі речі здадуться вам знайомими.

Коментарі

У Java, як і в C, існують однорядкові та блокові коментарі. Однак, крім цього, згідно з конвенцією Oracle, існують інші види коментарів: copyright-блок вгорі, doc-коментарі, TODO-коментарі, коментарі після statement`ів, коментарі для коментування коду. Ознайомтеся з прийнятими правилами використання коментарів, на захисті лабораторних вони вимагатимуться в обов'язковому порядку згідно з прийнятими конвенціями.

```java
/*
 * Copyright (c) 20121. Nick
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package ai142.ivanov;

/**
 * Це спеціальний коментар для документування (Doc comment).
 * За допомогою спеціальної утиліти (javadoc), такі коментарі можна перетворити на HTML-сторінки,
 * які разом створюють документацію до вашої програми.
 * Для зручності тут можна використовувати спеціальні посилання {@link Main} та <i>HTML-теги</i>
 */
public class Main {

    /**
     * Цей метод є "точкою входу" програми. У проекті може бути лише один метод із такою сигнатурою
     * @param args аргументи під час запуску програми
     */
    public static void main(String[] args) {

        // Це однорядковий коментар (single line comment)

        /*
         * Це коментар у вигляді блоку (block comment)
         */

        // TODO: це спеціальний TODO-коментар. Тут можна описати, що потрібно доробити на ділянці коду
        // FIXME: це теж TODO-коментар, зазвичай тут дрібні баги і що потрібно виправити на ділянці коду

        // TODO:2016-09-01:NickGodov: дописати виведення даних у файл

        int my_variable = 5; /* Коментарі після statement повинні бути вирівняні зліва */
        int my_other_variable = 999; /* за допомогою табуляції */

        // Якщо потрібно закоментувати код, то кожен рядок коментується однорядковим коментарем
// int my_old_variable = 100;
// int my_other_old_variable = 200;

        // Перед коментарем прийнято залишати порожній рядок
        int number = 924;
    }
}
```

Змінні

Змінні чутливі до регістру (var і Var – дві різні змінні), можуть бути нескінченної довжини, складатися з літер юнікод і цифр юнікод, символів _ і $.

Перший символ змінної може бути літерою, символом _ або $ (використовувати _ або $ Вкрай НЕ РЕКОМЕНДУЄТЬСЯ, вони існують для спеціальних ситуацій, які нас зараз не цікавлять, так що вважайте, що починатися змінна може тільки з літери юнікоду).

Вкрай не рекомендується використовувати літери національних алфавітів, кирилицю, трансліт. Тільки латинські літери, назви англійською. Також назви змінних не повинні співпадати зі списком зарезервованих слів:

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

Наведений нижче код дасть вам загальне уявлення про те, як треба називати змінні

```
public static void main(String[] args) {
    int spd = 25; // ПОГАНО: Можна, але з рекомендується, т.к. назва не інформативна
    int carminspd = 25; // ПОГАНО: Не заощаджуйте на назвах змінних!
    int carMinSpeed = 25; // ДОБРЕ: Назва змінної говорить сама за себе

    int s = 0; // МОЖНА: однолітерні допускаються, тільки якщо це якісь короткоживучі непрямі змінні

    int speed = 150; // ДОБРЕ: Нормально, зрозуміло, що змінна відповідає за швидкість
    int Speed = 150; // ПОГАНО: Вкрай не рекомендується, змінні не повинні починатися з капса
    int SPEED = 150; // ПОГАНО: Вкрай не рекомендується, повністю капсом пишуться константи

    const int MIN_SPEED = 25; // НЕ МОЖНА: Java const не використовується (хоча це зарезервоване слово)
    final int MIN_SPEED = 25; // ДОБРЕ: Java для констант використовується final

    int моя_змінна = 356; // ПОГАНО: Тільки латиниця
    int DŽDžDzóÁώÞЂ = 145; // ПОГАНО: Тільки латиниця
    int translit_epta = 29; // ПОГАНО: Трансліт – це повний моветон, тільки англійська!

    int $ myvar = 100; // ПОГАНО: Теоретично можна, але НЕ РЕКОМЕНДУЄТЬСЯ
    int _myvar = 100; // ПОГАНО: Теоретично можна, але НЕ РЕКОМЕНДУЄТЬСЯ
    int 2pac = 0; // НЕ МОЖНА: з цифри починати не можна
    int %d = 5; // НЕ МОЖНА: з інших знаків починати не можна
    int 'f' = 5; // НЕ МОЖНА: з лапок починати не можна

    // Якщо назва змінної складається із двох слів
    int max_speed = 150; // ПОГАНО: Використовувати _ відділення слів над константах не рекомендується
    int MaxSpeed = 150; // ПОГАНО: Вкрай не рекомендується, змінні не повинні починатися з великої літери
    int maxSpeed = 150; // ДОБРЕ: Ось так нормально, використовується lowerCamelCase
    final int MAX_SPEED = 150; // ДОБРЕ: Константи пишуться капсом, кожне слово відокремлюється _
}
```

Примітивні типи даних

У мові Java існують примітивні типи (аналогічні типу даних у C) і посилальні (або об'єктні) типи даних. На даний момент нас цікавлять лише примітивні типи даних.

Java – суворо типізована мова програмування. Це означає, що змінна, перед використанням, повинна бути оголошена і їй має бути присвоєний тип, який не можна

змінити. Також, при операціях присвоєння компілятор перевіряє відповідність типів (якогось механізму автоматичного приведення типів у Java немає).

Усього існують вісім примітивних типів даних: int, long, short, byte, double, float, char, boolean. Їх дуже легко запам'ятати: 4 типи для цілих чисел (коротке short, середнє int, довге long і байт), 2 типи для чисел з плаваючою комою (стара парочка double і float), 2 спеціальних типу - символ та булевий тип.

Type	Min	Max	RAM	Default	Оголошення та літерали
byte	-128	127	8 bit	0	`byte b = 100;`
short	-32,768	32,767	16 bit	0	`short b = 10000;`

int	-2^{31}	$2^{31}-1$	32 bit	0	`int a = 15;`
					`int aHex = 0xaa;`
					`int aBin = 0b0001111;`
					(це ж справедливо і для byte,short,long, якщо дотримуватися діапазонів)

long	-2^{63}	$2^{63}-1$	64 bit	0L	`long number = 10000L;`
double	4.9^{-324}	$\sim 1.8^{308}$	32 bit	0.0d	`double d = 6.6;`
float	$\sim 1.4^{-45}$	$\sim 3.4^{38}$	64 bit	0.0f	`float f = 5.5f;`

char	0	65535	16 bit	'\u0000'	`char c = ‘f’;`
					`char c = 63;`
					`char c = '\u2422';`

boolean	false	true	1 bit	false	`boolean b = true;`

Оператори розгалуження

Оператори розгалуження в C і Java практично ідентичні

```
private void ifStatements() {
    int a = 5;
    int b = 4;
    int min;

    // Так потрібно оформляти звичайний if
    if (a >= b) {
        min = b;
    }

    // Так потрібно оформляти if-else
    if (a >= b) {
        min = b;
    } else {
        min = a;
    }

    // Так потрібно оформляти if-else if-else
    if (a > b) {
        min = b;
    } else if (a < b) {
        min = a;
    } else {
        min = a;
    }

    // У Java використовується тернарний оператор
    min = (a> = b)? b: a;


    // Це рівнозначно наступному виразу
```

```
    if (a> = b) {
        min = b;
    } else {
        min = a;
    }

    // Так оформляється switch
    switch (a) {
        case 1:
            // щось робимо
            break;
        case 2:
            // робимо щось інше
            break;
        default:
            // це виконується в тому випадку, якщо жоден з кейсів не здійснився
            break;
    }
}
```

Цикли

Операції циклів також дуже схожі.

```
private void loops () {

    int progression = 0;
    // Так оформлюється for
    for (int i=0; i < 5; i++) {
        progression +=i;
    }

    // ПОГАНО: так оформляти цикли не рекомендується
    for (int i=0; i < 5; i++) progression +=i;

    // ПОГАНО: так оформляти цикли теж не рекомендується
    for (int i=0; i < 5; i++)
        progression +=i;

    // Порожній for
    for (int j = 0; j < 10; j ++);

    // Так оформляється while
    int iterator = 0;
    while (iterator < 10) {
        // робимо щось у циклі
        iterator++;
    }

    // Так оформляється do-while
    int loops = 10;
    do {
        // щось робимо
        loops--;
    } while (loops > 0);

    // Також, у Java є аналог foreach
    int[] array = {1, 2, 3, 4, 5};
    int sum = 0;
    for(int i : array) {
        sum += i;
    }

    // Цей же цикл можна уявити звичайним for`ом
    for(int i = 0; i < 5; i++) {
        sum + = array [i];
    }
}
```

Масиви

Робота з масивами Java трохи відрізняється від роботи з масивами в C, в основному, через механізм виділення пам'яті під масиви.

```
private static void arrays() {

    // Оголошення масивів

    /*
     * ДОБРЕ: згідно з усіма угодами за кодом та різними рекомендаціями, квадратні дужки
     * ставлять ПІСЛЯ ТИПУ ДАНИХ
     */
    int[] goodArray;

    /*
     * ПОГАНО: компілятор не видасть помилку, але такий синтаксис робить код менш читабельним
     */
    int badArray[];

    /*
     * НЕ МОЖНА: при оголошенні масиву не можна вказати його розмірність.
     * Java не виділить пам'ять, поки масив не буде ініціалізований
     */
    int[5] anotherBadArray;

    // Оголошення багатовимірних масивів

    int [][] 2DimensionalArray;
    int [][][] триDimensionalArray;

    // Ініціалізація масивів

    goodArray = new int[10]; // Ініціалізуємо масив із 10 елементами
    goodArray[0] = 15; // Привласнюємо значення першому елементу масиву
    goodArray[1] = 25; // Привласнюємо значення другому елементу масиву

    дваDimensionalArray = новий int [5][4]; // Двовимірний масив 5х
    2DimensionalArray[0] = new int[4];
    2DimensionalArray[1] = new int[8]; // ПОГАНО: Компілятор проковтне, але за фактом виділиться місце всього під 4
інти
    дваDimensionalArray[0][0] = 1; // Привласнюємо значення
    дваDimensionalArray[1][5] = 5; // НЕ МОЖНА: Компілятор видасть помилку

    System.out.print(twoDimensionalArray[1][6]); // НЕ МОЖНА: Компілятор видасть помилку

    // Оголошення з ініціалізацією

    int[] quickArray = {1, 2, 3, 4}; // Оголошуємо та відразу заповнюємо дані. Компілятор виділить місце під 4 інти.
    quickArray[5] = 6; // НЕ МОЖНА: Компілятор видасть помилку, т.к. індекс виходить за межі масиву

    int[][] quick2DArray = {
            {1,2,3},
            {1, 3, 4}
    };
}
```

Функції

Т.к. Java є об'єктно-орієнтованою мовою, функції тут називаються методами (на даний момент вони для нас ідентичні).

```
/*
 * Синтаксис функції:
 * [1-модифікатори доступу] [2-тип значення, що повертається] [3-ім'я]([4-аргументи]) [5-список винятків] {
 * 6- тіло функції
 * }
```

```
 *
 * 1 – модифікатори доступу: на даний момент вони нас не цікавлять, можна нічого не писати або писати private
 * 2 - тип значення, що повертається - тип даних або void, якщо функція нічого не повертає
 * 3 - обмеження як і на імена змінних, але є додаткові правила найменування, про них нижче
 * 4 - список аргументів через кому. Наприклад (int a, double b). Якщо немає аргументів – порожні дужки
 * 5 – виняток поки не розглядаємо, якщо їх немає, то просто нічого не пишуть
 * 6 - тіло функції, у ньому відбувається виконання функції. Якщо є тип даних, що повертається - повинен бути return
 */
private int findMinimum(int a, int b) {
    int min;

    min = (a < b)? a: b;
    return min;
}

/*
 * Назва методу починається з маленької літери, якщо кілька слів - використовується lowerCamelCase.
 * Перше слово має бути дієсловом (т.к. метод, як правило, "щось робить"), інші слова можуть бути
 * Прикметниками, іменниками тощо. Символ _ дуже бажано не використовувати (крім юніт-тестів)
 */

// ДОБРЕ: з маленької літери, перше слово - дієслово
private void drawCircle() {}

// ПОГАНО: Символи $ і _ не використовуємо
private void $er() {}

// ПОГАНО: я сказав "не використовуємо"!
private void draw_circle() {}

// ПОГАНО: перше слово з маленької літери
private void Draw() {}

// ПОГАНО: перше слово має бути дієсловом
private void circle() {}
```

ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ:

1. Напишіть метод, який приймає на вхід масив цілих чисел (довжиною 2 і більше) і повертає true, якщо в масиві кожен елемент більше або дорівнює попередньому. Інакше він повертає false.

2. Ви повинні реалізувати відому у Британії дитячу гру FizzBuzz. Напишіть метод, який виводить на екран числа від 1 до 100. При цьому замість чисел, кратних трьох, програма повинна виводити слово Fizz, а замість чисел, кратних п'яти - слово Buzz. Якщо число кратне і 3, і 5, програма повинна виводити слово «FizzBuzz».

3*. Реалізуйте метод, який приймає на вхід масив цілих чисел (довжиною 2 або більше) і повертає true, якщо масив можна розділити так, щоб сума чисел в обох частинах була рівною і false інакше. Наприклад:
method({1, 1, 1, 2, 1}) → true // {1, 1 ,1} = {2, 1}
method ({2, 1, 1, 2, 1}) → false // не виходить, т.к. {2, 1} < {1, 2, 1}, а {2, 1, 1} > {2, 1}
method ({10, 10}) → true // {10 = 10}

Для виведення даних у термінал використовуйте метод System.out.print(). Наприклад

```
private static void fizzBuzz() {
     int i = 5;

     System.out.print("Fizz"); // Виведе Fizz
     System.out.print("\n"); // новий рядок
     System.out.print("Buzz"); // Виведе Buzz
     System.out.print("Fizz" + "Buzz"); // Виведе FizzBuzz
     System.out.print(i); // Виведе 5
     System.out.print("Fizz" + i + "Buzz"); // Виведе Fizz5Buzz
}
```

ЩОБ УСПІШНО ЗДАТИ ЛАБОРАТОРНУ РОБОТУ, ВАМ НЕОБХІДНО:

1. Пред'явити робочий код;
2. Код має бути оформлений належним чином (назва base package, коментарі, назви змінних, відступи тощо);
3. Подати протокол, оформлений належним чином. Код повинен бути добре видно і структурований (моноширинний шрифт, код не повинен залазити на новий рядок). Див. нижче приклади хорошого та поганого протоколу.
4. Написати нормальні висновки.
5. Надати проект вихідного коду (архівом та посиланням в кінці протоколу на репозиторій).
6. Відповісти на питання викладача за матеріалом, написаним у протоколі, якщо такі будуть поставлені.
7. Розмістити усі результати роботи на своєму хмарному сховищу у каталогу «ПІБ – 2к – АД№Групи - ООП»

Цей протокол я прийму

```
/*
 * @(#)Blah.java 1.82 99/03/18
 *
 * Copyright (c) 1994-1999 Sun Microsystems, Inc.
 * 901 San Antonio Road, Palo Alto, California, 94303, U.S.A.
 * All Rights Reserved.
 *
 * This software is the confidential and proprietary information of Sun
 * Microsystems, Inc. ("Confidential Information"). You shall not
 * disclose such Confidential Information and shall use it only in
 * accordance with the terms of the license agreement you entered into
 * with Sun.
 */
```

```java
package ai14X.surname;

import java.blah.blahdy.BlahBlah;

/**
 * Class description goes here.
 *
 * @author Firstname Lastname
 * @version 1.82 18 Mar 1999
 */
public class Blah {
/* A class implementation comment can go here. */

    /**
     * classVar1 documentation comment
     */
    public static int classVar1;

    /***
     * classVar2 documentation comment that happens to be
     * more than one line long
     */
    private static Object classVar2;

    /**
     * instanceVar1 documentation comment
     */
    public Object instanceVar1;

    /**
     * instanceVar2 documentation comment
     */
    protected int instanceVar2;

    /**
     * instanceVar3 documentation comment
     */
    private Object[] instanceVar3;

    /**
     * ...constructor Blah documentation comment...
     */
    public Blah() {
        // ...implementation goes here...
    }

    /**
     * ...method doSomething documentation comment...
     */
    public void doSomething() {
```

```
        // ...implementation goes here...
    }

    /**
     * ...method doSomethingElse documentation comment...
     *
     * @param someParam description
     */
    public void doSomethingElse(Object someParam) {
        // ...implementation goes here...
    }
}
```

А такий ні

```
package ai14X.surname;
import java.blah.blahdy.BlahBlah;
public class Blah {
public static int classVar1;
private static Object classVar2;
public Object instanceVar1;
protected int instanceVar2;
private Object[] instanceVar3; public Blah() {
      // ...implementation goes here...
   }
public void doSomething() {
}
public void doSomethingElse(Object someParam) {
      // ...implementation goes here...
}}
```