Материалы

1. Книги
    a. Брюс Эккель, Филосовия Java
    b. Студенческий конспект лекций похоже курса java
2. Видео некоторых лекций
3. Те же видео из ГУАП на канале кафедры БИС

Семестр - I

Требования:

1. **При выполнении первых трех лаб нельзя пользоваться IDE.**
2. **Стиль оформления исходников: Java Style Guide**
3. **Никаких публичных полей (использовать сеттеры и геттеры)**

Задание 0. Сортировка (не нужно делать).

Отсортировать с выписыванием промежуточных шагов массив из 16 элементов:

1. простой сортировкой (показать первые 5 шагов)
2. быстрой сортировкой (показать все шаги)
3. сортировкой слиянием (показать все шаги)т

Задание 1. Int.

Напишите класс, реализующий «правильные с объектно-ориентированной точки зрения целые числа”, В классе должны быть определены

- методы increment() и decrement(), соответственно увеличивающие и уменьшающие число на 1;
- методы add(Int n) и substract(Int n), увеличивающие и уменьшающие число на n;
- метод toString()

Примечание 1: в методах add и substract передаются значения типа Int (с большой буквы), а не int.
Примечание 2: для сдачи основного задания нельзя придумывать свои методы и конструкторы, нужно использовать указанные выше.
Примечание 3: применить инкрементирование 1000 раз - плохой вариант.
Примечание 4: в доп. задании можно создать конструктор Int(int value).

При создании нового объекта должно создаваться число, равное 0.

Напишите наиболее короткую программу, которая используя только класс Int, выводит на экран число 1000. Программа должна быть чисто объектно-ориентированной. В частности, в ней нельзя использовать оператор присваивания.

Задание 2. Матрица.

Напишите класс Matrix, реализующий квадратные матрицы. В нем должны быть определены

- конструктор с целочисленным параметром --- размером матрицы, создающий единичную матрицу;
- методы **Matrix** sum(**Matrix**) и **Matrix** product(**Matrix**), вычисляющие сумму и произведение матриц
- методы setElement(int row, int column, int value) и getElement(int row, int column), для обращения к элементам матрицы;
- метод toString() *(Примечание: необходимо использовать StringArray или StringBuilder).*

Во всех методах предполагается, что передаваемые параметры всегда корректны.
Напишите программу, выводящую **первые 10** степеней матрицы:

```
[1 1]
[1 0]
```

Задание 3. Наследование.

Напишите класс Matrix, реализующий матрицы и расширяющий его класс SquareMatrix, реализующий квадратные матрицы. В классах должны быть определены:

- конструкторы с параметрами размерами матриц, создающие нулевую матрицу для Matrix и единичную для SquareMatrix;
- методы **Matrix** sum(**Matrix**) и **Matrix** product(**Matrix**), вычисляющие сумму и произведение матриц (метод sum должен быть переопределен в SquareMatrix);
- методы setElement(int row, int column, int value) и getElement(int row, int column), для обращения к элементам матрицы;
- метод toString().

Напишите собственный класс исключения, расширяющий *(наследующий)* класс RuntimeException. Во всех конструкторах и методах должны бросаться исключения в тех случаях, когда соответствующая операция невозможна (например, при сложении матриц разных размеров). Исключения должны содержать информацию о том, какая именно проблема возникла. Достаточно хранить эту информацию в виде строки, возвращаемой методом getMessage().

Примените к написанной программе:

1. разложите классы по пакетам.
2. напишите слово **final** в тех случаях, где оно разумно.
3. реализуйте для матриц метод equals().

Задание 4. LinkedList.

Напишите класс SortedIntegerList, который хранит отсортированный в порядке возрастания список целых чисел. Внутри класса список должен храниться с помощью LinkedList. У SortedIntegerList должны быть определены:

- Конструктор с булевским параметром:
    - если этот параметр принимает значение true, то в создаваемом списке разрешены повторяющиеся элементы;
    - false — запрещены повторяющиеся элементы.
- Методы add(int) — добавление числа в список; и remove(int) — удаление числа из списка:
    - если добавление/удаление невозможно — метод не делает ничего;
    - операции добавления/удаления должны требовать **не более чем одного** прохода по списку;
- Метод equals();

    *Примечание: использовать **везде** итератор.*

Напишите программу, проверяющую работу класса SortedIntegerList. Постарайтесь реализовать возможно полный набор проверок.

Задание 5. Интерфейс.

Напишите интерфейс IMatrix с несколькими реализациями:

- UsualMatrix и расширяющий его SquareMatirx из предыдущих заданий;
- SparseMatrix — для разреженных матриц. Должен быть реализован с помощью LinkedList (возможно, вам потребуется создать какие-то дополнительные классы, которые должны быть вложенными/внутренними).

Все общие методы должны быть представлены в интерфейсе IMatrix.

Напишите программу, создающую 2 случайные матрицы размером 1000x1000 с 1000 ненулевых элементов в каждой двумя способами: с помощью обычных и разреженных матриц. Проверьте, что сложение и умножение для разных видов матриц дает одинаковые результаты.

Задание 6. Ввод-вывод.

Нужно реализовать две небольшие программы.

1. Реализовать класс FormattedInput с двумя статическими функциями:
Object[] scanf(String format). Читает с System.in.
Object[] sscanf(String format, String in). Читает из строки in.

format --- строка со спецификацией формата ввода (может быть несколько спецификаторов в одной строке, например, «%d %d %f»). Список спецификаторов:
%d --- целое int
%f --- дробное double
%s --- строка
%c --- символ

Если ввод пользователя не соответствует спецификации, то функция запрашивает ввод повторно.
Пример:

```
main(..) {
……….....
   Object vals[] = scanf(“%d %s %c %f”);
……..
}
```

Ввод пользователя: 10 ten v 11.2

2. Реализовать программу EncodingConverter для перекодирования текстовых файл из одной кодировки в другую. Программа должна получать параметры из командной строки и контролировать их корректность.

Пример вызова: java EncodingConverter in.txt out.txt utf8 cp1251

Задание 7. Map.

Реализуйте класс для хранения настроек Settings, в котором хранятся пары «имя параметра, значение». «Имя параметра» задается строкой, а «значение» целым числом. Реализация должна использовать класс HashMap. В классе Settings должны быть определены:

- toString() и equals()
- put(String, int)
- int get(String)

- delete(String)
- loadFromBinaryFile(String filename)
- saveToBinaryFile(String filename)
- loadFromTextFile(String filename)
- saveToTextFile(String filename)

Задание 8. Thread.

Реализовать класс ParallelMatrixProduct для многопоточного умножения матриц UsualMatrix. В конструкторе класс получает число потоков, которые будут использованы для перемножения (число потоков может быть меньше, чем число строк у первой матрицы).

В функции main сравнить время перемножения больших случайных матриц обычным и многопоточным способом. Получить текущее время можно с помощью методов класса System.

Задание 9. Synchronized.

Написать программу, приводящую к ситуации взаимной блокировки (deadlock).

В дополнительных заданиях ничем кроме synchronized из примитивов синхронизации пользоваться нельзя.

Задание 10. Читатели-писатели.

Написать класс стек на основе связного списка (классом LinkedList пользоваться нельзя) с методами

- модифицирующими: void push(int), int pop()
- читающими: equals, toString

Класс должен быть пригоден к использованию в многопоточных программах. Написать две реализации:

- SynchroStack --- со стеком одновременно работать может только один поток
- SynchroStackFast --- со стеком одновременно может работать либо один модифицирующий поток, либо несколько читающих потоков

В main с помощью замеров времени показать преимущество SynchroStackFast при работе в многопоточной программе.

Семестр - II

1. Все лабы необходимо сдавать в ОС Linux. Мы рекомендуем Ubuntu в виртуальной машине (Virtualbox, инструкция) и на отдельном разделе. Рекомендуемая книга “Основы Linux от основателя Gentoo” “Основы Linux от основателя Gentoo” .
2. Первые две лабы --- лабы 8 и лаба 9 из первого семестра.
3. Примеры с лекций

Задание 11. Чат для двух пользователей.

Написать текстовый чат для двух пользователей на сокетах. Чат должен быть реализован по принципу клиент-сервер. Один пользователь находится на сервере, второй --- на клиенте. Адреса и порты задаются через командную строку: клиенту --- куда соединяться, серверу --- на каком порту слушать. При старте программы выводится текстовое приглашение, в котором можно ввести одну из следующих команд:

1. Задать имя пользователя (@name Vasya)
2. Послать текстовое сообщение (Hello)
3. Выход (@quit)

Принятые сообщения автоматически выводятся на экран. Программа работает по протоколу UDP.

Задание 12. Многопользовательский чат.

Написать текстовый многопользовательский чат.

1. Пользователь управляет клиентом. На сервере пользователя нет. Сервер занимается пересылкой сообщений между клиентами.
2. По умолчанию сообщение посылается всем участникам чата.
3. Есть команда послать сообщение конкретному пользователю (@senduser Vasya).

Программа и все дополнительные задания должны работать по протоколу TCP

Задание 13. Сервлеты.

Реализовать сервлет для работы с записной книжкой. В записной книжке для каждого человека хранится его имя и список телефонов (их может быть несколько). При старте сервлет загружает записную книжку из текстового файла. Сервлет должен позволять:

1. Просматривать список записей
2. Добавить нового пользователя
3. Добавить новый телефон

На главной странице сервлета находится список записей. Вверху страницы ссылки --- добавить. Каждая из ссылок ведет на отдельную страницу, где с помощью элементов <input type="text" name="username" /> в форме вводятся необходимые данные. Для отправки данных сервлету есть кнопка submit.

В качестве контейнера сервлетов рекомендуется использовать либо сервер Tomcat, либо сервер Jetty

NB: Синхронизация при работе нескольких пользователей с одной записной книжкой.

Задание 14. Функциональная спецификация на курсовик.

Доп. задания нет. Материалы по LaTeX

Требования:

1. Должна содержать подробное описание, список возможностей программы и инструкцию пользователя (объем --- 1 или 2 страницы)
2. Должна быть написана в LaTeX

Задание 15. HttpSession. CSS.

Реализовать сервлет для организации доски объявлений. Объявление содержит текст и заголовок (время размещения и имя пользователя).

Для того, чтобы разместить, необходимо ввести логин и пароль (пройти аутентификацию). При старте сервлет загружает базу пользователей и их паролей из текстового файла. Просматривать объявления можно без аутентификации (ввода логина и пароля).

На главной странице находится ссылка “войти в систему” и показывается список объявлений. После входа в систему добавляются ссылки: “выйти из системы”, “добавить объявление”.

1. Для аутентификации необходимо использовать класс HttpSession
2. Для перенаправления пользователя на другую страницу и включения в страницу готов кусков html кода можно воспользоваться классом RequestDispatcher
3. Протестировать, что при очистке cookie в браузере, пользователь выходит из системы.

Между перезагрузками сервера (рестарт Tomcat) список объявлений можно не сохранять.

Дополнительные требования (CSS):

1. Оформление страницы должны быть задано в отдельном файле style.css
2. Должны быть созданы три стиля:
    a. Заголовок объявления
    b. Текст объявления
    c. Ссылка с командой (“войти в систему”, “выйти из системы”, “добавить объявления”)
3. В работе должно быть два варианта оформления (переключение --- переименованием файлов на сервере)

Задание 16. Javascript.

Разработать сервлет, который показывает двухуровневый список. Список загружается из текстового файла на сервере, который имеет следующий формат:

```
* Звери
   * Волк
   * Корова
* Птицы
   * Курица
   * Орел
   * Попугай
```

Элементов первого и второго уровня может быть произвольное количество. Элементы списков второго уровня смещены на 4 пробела вправо.

Реализовать интерфейс для удобной работы со списком на javascript (js). При выводе в виде html список верхнего уровня становится нумерованным. Около каждого элемента первого уровня выводится символ, при нажатии на который скрываются или показываются элементы соответствующего списка второго уровня. Если список второго уровня показан, то выводится символ "-", позволяющий скрыть список. Если список второго уровня скрыт, то выводится символ "+", позволяющий показать список.

Исходное состояние:

```
1. Звери [+]
2. Птицы [+]
```

Состояние после нажатие на "+" около пункта Птицы:

```
1. Звери [+]
2. Птицы [-]
   * Курица
   * Орел
   * Попугай
```

При скрытии и показе списка не должно быть запросов на сервер. Графический интерфейс обеспечивает программа на js, выполняющаяся внутри браузера.

Курсовые работы

Требования к отчетности (на любую оценку):

1. Перед началом выполнения курсовика необходимо написать функциональную спецификацию (список возможностей программы) и согласовать ее с преподавателем. Засчитывается как отдельная лаба.
2. Проект должен включать отчет в TeX. Структура отчета:
    a. титульный лист
    b. функциональная спецификация (словесное описание того, что планировалось сделать, т.е. развернутая постановка задачи)
    c. руководство пользователя со скриншотами
    d. описание архитектуры программы; архитектура --- из каких частей состоит программа и как эти части взаимодействуют (графическая схема с пояснениями);
    e. описание наиболее важных классов --- про каждый класс описать его назначение, а также описать назначение наиболее важных публичных методов (листинги реализации методов приводить не нужно, достаточно описаний и сигнатур)

Требования к исходному и исполняемому коду (более высокая оценка добавляет дополнительные требования):

1. На любую оценку
    a. В общем случае нельзя использовать сторонние библиотеками (исключения нужно обсуждать с преподавателем).
    b. Архитектурно проект должен состоять из нескольких подсистем. Наиболее типичное решение: Model-View-Controller (при необходимости View и Controller можно совмещать).
    c. Код должен соответствовать Java Coding Convention
    d. Под базой данных понимается текстовый файл (использование СУБД на усмотрение студента)
2. На оценку 3:
    a. запускаться из командной строки
3. На оценку 4:
    a. проект должен быть разложен по пакетам
    b. исполняемый код должен быть упакован в jar (для сервлетов не нужно)
    c. проект должен включать unit-тесты (на модель)
4. На оценку 5:
    a. проект должен лежать на github

b. проект должен собираться либо с помощью Maven, либо с помощью Gradle (компиляция, запуск тестов, упаковка тестов, очистка)

По каждому курсовику преподаватель практики и студент договариваются о:

- Функциях, доступных пользователю в базовой версии (оценка 4)
- Функциях, доступных пользователю в продвинутой версии (оценка 5)

Эти списки функций должны быть зафиксированы в функциональной спецификации.

На сокетах:

- Все серверные программы должны писать в лог (текстовый файл) о своих действиях (пример: подсоединился клиент с адреса X, клиент X загрузил файл Y)

1. Многопользовательский Paint. Несколько пользователей могут одновременно рисовать на общей доске. Возможные опции:
    a. Поддержка нескольких досок на сервере.При соединении с сервером пользователь выбирает: начать новую доску или присоединиться к существующей.
    b. Сервер хранит содержимое досок. Пользователь может присоединиться к уже открытой доске и увидеть ее текущее состояние.
    c. Можно рисовать и стирать.
2. TFTP клиент и сервер.
    a. Программа клиент может быть либо с графическим, либо с текстовым интерфейсом
    b. Необходимо проверить, что клиент совместим со сторонним сервером
    c. Необходимо проверить, что сервер совместим со сторонним клиентом
3. Программа для автоматической синхронизации файлов в двух каталогах на разных компьютерах. Возможные опции:
    a. Клиентская синхронизация: синхронизация между двумя компьютерами. Каждый из двух экземпляров программы получает адрес другого экземпляра и каталог для синхронизации. Недостающие файлы должны быть скопированы. Если два файла имеют одинаковое имя, но разное содержимое (можно проверить с помощью хэш-функции), то программа пишет в лог сообщение о конфликте.
    b. Серверная синхронизация: аналог Dropbox с множеством клиентов. Каждый клиент получает адрес сервера и каталог для синхронизации. Весь обмен данными происходит через сервер.
4. Программа, которая автоматически скачивает заданный набор веб-страниц и строит по ним поисковый индекс. Поиск по поисковому индексу через командную строку.
    a. Есть две программы: паук и поисковик. Паук получает список веб страниц в виде текстового файла и строит по ним поисковый индекс, который записывается в текстовый файл. Поисковик на вход получает из командной строки запрос в виде списке ключевый слов, а на выход выдает

ранжированный список из веб страниц (в виде HTML файла), ранее загруженных пауком. Больший ранг в списке имеют веб страницы, релевантные запросу.
    b. Для построения поискового индекса каждая веб страница разбирается на слова (рекомендуется для простоты работать со страницами на английском языке). В каждой странице находятся слова, которые встречаются на ней чаще всего (исключая предлоги). Пять самых частых слов будем считать ключевыми словами, описывающими эту страницу.
    c. Поисковый индекс состоит из строк вида: имя веб страницы, список ее ключевых слов, частоты ключевых слов.
    d. Наиболее релевантной к запросу будем считать страницу, у которой максимальное количество ключевых слов пересекается со словами запроса. Также можно учитывать частоты ключевых слов.
5. Сетевая игра в крестики-нолики на поле 10x10.
    a. Поддержка нескольких досок на сервере. При соединении с сервером пользователь выбирает: начать новую доску или присоединиться к существующей, если на ней только один игрок.
    b. Программа может быть как с текстовым, так и с графическим интерфейсом.
6. Сетевая игра “теннис” (необходимо знание Swing).
    a. Две полоски с двух сторон корта --- игроки. Игрок может перемещаться вверх или вниз. Корт сверху и снизу окружен кирпичной стеной. Мячик отскакивает от стены с полным сохранением импульса.
    b. Поддержка нескольких игр на сервере. При соединении с сервером пользователь выбирает: начать новую игру или присоединиться к существующей, если н i’ma ней только один игрок.
    c. Рекомендуется использовать UDP
7. Сетевая игра “морской бой”
    a. Поддержка нескольких игр на сервере. При соединении с сервером пользователь выбирает: начать новую игру или присоединиться к существующей, если на ней только один игрок.
    b. Программа может быть как с текстовым, так и с графическим интерфейсом.

На HTTP (сервлеты):

- Нужна аутентификация на cookies.
- Информация хранится в базе данных (текстовый файл).
- Для хранимых сущностей (дела, тикета, товары) должна быть возможность сделать CRUD (create, read, update, delete)
- Должна быть поддержка вывода в браузере сущностей, занимающих больше чем одну страницу (много дел, много товаров, много тикетов)

1. Web-каталог для магазина (с корзиной). Возможные опции:
    a. Учетная запись продавца, который может наполнять магазин товарами

    b. Учета количества товаров. Если пользователь сложил последний экземпляр товара в свою корзину, то товар пропадает из каталога.
2. Список дел (to-do list). Возможные опции:
    a. Можно иметь несколько списков
    b. Списки могут быть вложенными
    c. Списки можно расшаривать между несколькими пользователями
3. Баг-треккер
    a. Можно создать тикет с параметрами: баг или фича, статус (открыт или закрыт), кто отвечает
    b. Есть два списка: открытые и закрытые тикеты. Если у тикета сменить статус, то он переходит в соответствующий список
    c. После создание все поля списка можно редактировать
4. Web-каталог статей с поиском по автору, теме и ключевым словам
    a. Пользователь можно загрузить в свой каталог на сервере pdf статью и заполнить для нее библиографические данные (авторы, название, ключевые слова год)
    b. Пользователь может скачать статью из своего каталога
    c. Пользователь может искать статью по библиографическим данным
5. Электронная ведомость
    a. Должно быть два типа пользователей: преподаватель (может редактировать) и студент (может только просматривать)
    b. Можно иметь несколько ведомостей под разные предметы
    c. Можно динамически задавать количество строк и столбцов
6. Система для обмена файлами через веб.
    a. Можно загружать файлы в каталог и скачивать из каталога
    b. Можно расшаривать файлы другим пользователям.

Простые курсовики (на 3):

1. Пинг с поддержкой одного клиента
    a. Клиент посылает запрос серверу и запоминает время отправки.
    b. Клиент получает ответ от сервера и запоминает время получения
    c. Клиент выводит RTT до сервера. Если пакет не вернулся через 1 минуту, то клиент выводит, что сервер недоступен.
2. Пересылка файла на другой компьютер
    a. Клиент получает имя файла при старте и посылает файл клиенту
    b. Сервер сохраняет полученный файл в заданный каталог
3. Сервлет для чата с ручным обновлением
    a. Клиент может послать серверу сообщения
    b. В ответ сервер отвечает клиенту всем списком уже имеющихся сообщений
    c. Можно вручную получить с сервера список последних сообщений, нажав “обновить” в браузере
4. Сервлет для мудрых цитат
    a. На сервере хранится списком цитат в текстовом файле

    b. В ответ на запрос клиента сервер присылает из этого списка случайную цитату
5. Сервлет для “смишных картинок”
    a. На сервере хранится каталог “смишных картинок”
    b. В ответ на запрос клиента сервер присылает случайную картинку
6. Сервлет для просмотра списка файлов на сервере
    a. Клиент присылает имя каталога, сервер отвечает списком
7. Сервлет записной книжки
    a. Создать сервлет, который генерирует записную книжку со случайным содержимым. На стартовой странице можно указать количество записей и формат вывода xml или json.
8. Сервлет для форматированного отображения файлов
    a. Создать сервлет, который выдают один из текстовых файлов из каталога (задается) в виде html страницы с заданным выравниванием, шрифтом, размером шрифта, цветом фона и цветом текста (указывается на стартовой странице). Необходимо использовать css.
9. Записная книжка на JSP
    a. Переделать пример из лекций про записную книжку под технологию JSP. Начальное заполнение книжки должно считываться из файла.
10. Сервлет для просмотра гит репозитория
    a. Создать сервлет, который позволяет просматривать гит репозиторий, находящийся на сервере с помощью команды git log. Репозиторий создан и заполнен заранее. На стартовой странице можно указать формат вывод (краткий или полный) и глубину просмотра.

Spring / SpringBoot (на 5):

Помимо требований на 5, представленных выше, необходимо выполнить следующие требования.
Обязательные:

- Разделение на пакеты,как принято в Spring (service,controller,repository,model, etc.)
- Реализация через интерфейсы (можно легко подменить одну имплементацию сервиса/репозитория на другую и проект продолжит работать)
- Работа с данными - Spring Data Jpa(например Hibernate) или JDBC
- Frontend с помощью JSP/Thymeleaf (можно предложить своё)
- Тестирование Junit (для repository, service и др)

Дополнительно:

- Для тестирования использовать Mockito
- Авторизация с помощью Spring Security, JWT
- упаковать приложение (или базу данных) в Docker

Темы курсовых можно брать из списка тем для сервлетов или предлагать свои на согласование с преподавателем.

Производственная Практика (2ой курс)

Орг вопросы:

1. Тема -- графическая программа на swing (java).
2. Длительность --- 1 месяц, начало: 30 июня
    a. Будет проведено 2-3 лекции (графические библиотеки, swing, проектирование графической программы)
        i. Графические библиотеки, введение в Swing,
            1. Видео. Часть I.
            2. Видео, Часть II
        ii. Назначены дни вопросов и приема результатов (2 раза в неделю)
    b. В 2025: проходит удаленно через telemost.yandex.ru
3. Допускается прохождение практики в какой-нибудь фирме. Для этого:
    1. До начала (заранее!) практики фирма и ГУАП должны подписать договор о прохождении практики (со стороны ГУАП договорами занимаются Н. В. Марковская)
    2. После 2022: все вопросу надо обсуждать в Н. В. Морковская
    3. В 2022: в конце практики прислать ответственному за практики (22 -- Линский) отзыв (1 лист):
        a. 1 абзац -- описание задачи (должно быть про программирование)
        b. рекомендуемая оценка
        c. ФИО и подпись руководителя в фирме
        d. Печать фирмы

Предварительная работа:

1. Выбрать тему и сложность, послать тему и сложность письмом Линскому Е. М. (evgeny.linsky@guap.ru). Тема будет закреплена за вами в ведомости.
2. Написать функциональную спецификацию (1-2 страницы) сложности 2 и 3
    a. зачем нужна программа?
    b. что должна делать программа с точки зрения пользователя
        i. скетчи будущего интерфейс (можно картинка от руки, сфотографированная на телефон)
        ii. инструкция пользователя: "чтобы создать родственную связь нажмите на кнопку", "элементы, обозначающие родственников, можно перемещать по экрану"
3. Послать спецификацию в формате PDF письмом Линскому Е.М (evgeny.linsky@guap.ru). Если спецификация зачтена, то в ведомости тема будет покрашена в желтый цвет

Отчет:

1. Для того, чтобы практика была зачтена, необходимо сдать программу и код, а также загрузить отчет в электронный кабинет (дисциплина “производственная”)
2. Если отчет принят, то оценка будет окрашена цветом и выставлена в бумажную ведомость
3. Шаблон отчета можно найти тут
    a. В титуле необходимо заполнить своими данными пункты, помеченные желтым (в пункте исходные данные -- книга или ссылка про Swing)
    b. Для проходивших в фирме: содержимое отчета -- отзыв из фирмы (просто вставить за титулом)
    c. Для проходивших у меня (Линского): содержимое отчета -- инструкция пользователя, т.е. похоже на функциональную спецификацию, но должно быть описано, что фактически получилось в итоге, вставлены реальные скриншоты вместо эскизов. Если сдавали лабы – все три лабы со заданиями, доп заданиями и скриншотами в одном отчете.

Сложность 1:

1. Сделать программу, в которой нарисован домик
2. Сделать программу “доска для рисования”. Если зажать кнопку мыши и водить курсором по окну, то в окне рисуется траектория точек.
3. Сделать программу с диалогом “данные о человеке”. Необходимо использовать один из менеджеров размещения. При нажатии на кнопку OK в диалоге должен появиться новый информационный диалог, в котором введенные данные написаны через запятую. Должны быть следующие поля: имя (text box), фамилия (text box), пол (radio button), страна (combobox), возраст (text box).

Сложность 2 и 3:

1. Пакеты (model, view, io, test)
    a. model: FamilyTree, Person
    b. view: MainFrame, FamilyView, PersonInfo
    c. io: FamilyReader, FamilyWriter
    d. test: FamiyTreeTest, FamilyReaderTest, TestRunner
2. Тесты:
    a. для публичных методов FamilyTree
    b. для публичных методов FamilyReader
        i. корректный файл
        ii. некорректный файл

Типовые ошибки:

1. Не учтены архитектурные требования
    a. Нет пакетов (model, view, io)
    b. Нет Model-View (например, в model импорты из swing/awt/view)
    c. Нет тестов
    d. На 5 тесты не на junit

2. Наличие публичных полей у классов (допускается только public static final для констант)
3. Есть сохранение игры в файл, но нет загрузки
4. Абсолютные пути к файлам (C:\user\student\wall.jpg) в программе или тестах
5. При поломке model (пошли в модель и сделали логическую ошибку в программе) не падают тесты

Сложность 2:
Программа для просмотра генеалогического дерева семьи (дети, родители, супруги). Число поколений неограничено. Программа загружает из текстового файла информацию о родственных связях и рисует их графически. При клике на узел дерева выводится диалог с подробной информацией о соответствующем члене семьи.

Сложность 3:
1. Есть возможность редактирования положения узлов на экране (перетаскивать узлы по экрану с перерисовкой всех связей между ними).
2. Есть возможность добавления/удаления членов семьи.
3. Есть возможность загрузки/сохранения из файла (можно указать имя файла через стандартный диалог). Вызов этих функций должен быть либо через меню, либо через панель инструментов).
4. Тесты должны быть реализованы через библиотеку junit (версии 3 или 4).

Примеры индивидуальных заданий (подробности нужно согласовывать лично). Возможен свой вариант задания.
1. Логические (или булевы) схемы используются в математике и технике для формализации манипуляций с двоичными данными.Схема состоит из логических элементов (например, НЕ, И, ИЛИ), соединенных проводами. У каждого элемента есть входы, на которые подаются данные) и выходы (с которых считывается результат). Например, у элемента И два входа и один выход, причем значение на выходе истинно только если истинны значения на обоих входах. Соединяя выходы одних элементов с входами других, можно построить схему для вычисления любой функции.

   Задача: написать программу, позволяющую создавать схемы, подавать на их входы данные и получать результат.

2. Игра с искусственным интеллектом и графическим интерфейсом
   a. Варианты
      i. Хексагон
      ii. Шашки
      iii. Нарды
      iv. Го

   b. На 4
      i. игра человека с человеком
      ii. сохранение партии в файл
   c. На 5 (все что на 4)
      i. игра с компьютером
      ii. история ходов с возможностью откатиться на предыдущий ход
3. Редактор кроссвордов. Программа позволяет:
   a. Редактировать сетку кроссворда
   b. Сохранять и загружать сетку из файла
   c. Заполнять сетку автоматически с помощью словаря

3. Игра двухмерный платформер
   a. На 4
      i. Несколько уровней
      ii. Таблица рекордов, которая хранится между запусками
   b. На 5 (все что на 4)
      i. Редактор уровней (сделанный уровень можно подключить к игре)

4. Personal Task Manager. Приложение отображает список задач на специальной временной диаграмме, позволяющей визуально оценить общий объем и длительность задач. Продукт представляет собой Personal Task Manager (PTM). Приложение отображает список задач на специальной временной диаграмме, позволяющей визуально оценить общий объем и длительность задач.

   В данной версии программы реализованы следующие возможности:
- Создание, удаление задачи и изменение ее параметров (с помощью диалога);
- Сохранение задач в XML-файл при каждом изменении;
- Изменение масштаба временной диаграммы (количества отображаемых дней), переключение между режимами «3 дня» и «Неделя»;
- Перемещение по временной диаграмме на один день назад/вперед или сразу 3 или 7 дней в зависимости от масштаба;
- Переход в сегодня (то есть к такому виду, когда текущий день находится в середине интервала отображаемых дат) специальной кнопкой. Границы «сегодня» выделены особым цветом, все, что за ним, – прошлое – «закрашено» серым.

5. Умный крот. Логическая игра об умном кроте, которому нужно переместить все кучки зерна к мешкам. Игра ведётся на поле, окруженном стеной. На поле находятся преграды – стены, кучки зерна и пустые мешки, в которые нужно перетащить зерно. Умный крот может передвигать только один ящик и только перед собой. Сквозь стены ходит не может. При попадания кучки зерна в мешок, её можно сдвинуть. Когда всё зерно окажется в мешках – вы можете праздновать победу. За каждое действие, в том числе и отменить/повторить, прибавляется число сделанных шагов. Очки вычисляются как процент от минимального необходимого количества шагов. Если полученный результат больше

минимального в таблице лучших результатов, то ваше имя окажется в этой таблице (если вы его, конечно же, введёте). Управление - клавишами курсора.

6. Bantumi. Bantumi (Kallah) — древняя африканская игра. Каждому игроку принадлежит 6 малых горшков вдоль его длинной стороны поля и один большой горшок по его правую руку, называемый каллахом. В начале игры в каждый горшок помещается некоторое количество камней (в разных вариантов правил от 3 до 6). Цель игры состоит в том, чтобы в своём каллахе камней оказалось больше, чем в калахе противника. В процессе хода игрок выбирает свой малый горшок с камнями, забирает оттуда все камни и раскладывает их по одному в остальные горшки, двигаясь против часовой стрелки. Первый камень кладётся в горшок справа от выбранного, затем в следующие в том же направлении, включая свой каллах и малые горшки противника, но исключая его каллах. Если последний камень был помещён в свой каллах, то игрок ходит повторно. Если последний камень помещён в свой малый горшок, в котором нет камней, то этот камень и камни из горшка напротив, принадлежащего противнику, перекладываются в свой каллах. Такой ход называется захватом, после которого право хода передаётся другому игроку. Если у игрока, получившего очередь хода, не осталось ни одного камня в малых горшках, то игра немедленно прекращается и все камни противника попадают в каллах противника. И очевидно, что если в одном из каллахов оказалось больше половины общего числа камней, то игрок, которому принадлежит этот каллах, победил.

7. Graph Editor

Программа предназначена для работы с основными типами графов (ориентированными, неориентированными, взвешенными и невзвешенными).
В данной версии программы реализованы следующие возможности:

- сохранение графа в файл и загрузка из файла (граф хранится в формате xml)
- создание нового графа
- редактирование внешнего вида графа (добавление и удаление вершин и ребер, изменение веса ребра, перенос ребра)
- отмена/повтор последних действий
- конвертация различных типов друг в друга
- различные укладки графа (круговая, топологическая, в виде минимального остовного дерева)
- реализованы несколько алгоритмов на графе
    - Топологическая сортировка (поиск цикла)
    - Минимальное остовное дерево
    - Алгоритм Дейкстры
    - Поиск максимального потока методом Эдмондса-Карпа

План лекций

Семестр 1:
TBD

Семестр 2:

1. Введение в сетевое программирование
    a. OSI: PHY, MAC, Network (IP), Transport (TCP, UDP), Session + Presentation (TLS), Application
    b. Пакет, заголовок, IP адреса, порты
    c. Архитектуры:
        i. 2 пользователя: клиент --- клиент + сервер (морской бой двух игроков)
        ii. Много пользователей: много клиентов --- сервер (многопользовательских чат)
    d. Необходимость многозадачности в сетевых приложениях
2. Многопоточность в java
    a. Потоки в Java (класс Thread)
    b. Методы синхронизации в java (synchronized)
3. Клиент-серверное приложение на java (два пользователя)
    a. Абстракция сокет. Серверный сокет, клиентский сокет
    b. UDP сокет, TCP сокет
    c. UDP сокет, программирование
    d. TCP сокет, программирование
    e. Многопоточность для двух пользователей: интерфейс и сеть
4. Клиент-серверное приложение на java (много пользователей)
    a. Многопоточность в многопользовательском приложении: обслуживание нескольких клиентов одновременно
    b. Многопользовательский сервер, программирование
5. Веб-сайт
    a. Статический веб-сайт
    b. Браузер, веб-сервер
    c. Введение в HTML.
    d. Введение в HTTP.
6. Сервлеты.
    a. Динамический веб-сайт, CGI, сервер приложений
    b. Библиотеки и фреймворки (фреймворк servlets)
    c. Сервер приложений Tomcat. Пример конфигурации.
    d. Сервлеты, программирование
    e. Cookie
7. Динамический HTML.

a. Введение в javascript
b. Динамический javascript и HTML

Темы для практик по курсовому проектированию:

1. Системы контроля версий git
2. Сборка приложений Ant, Maven
3. XML
4. Юнит-тесты (junit 3 и 4 по выбору)