ВЕБ-ТЕХНОЛОГИИ

Разминка с Java

OlgaSmolyakova

ВВЕДЕНИЕ

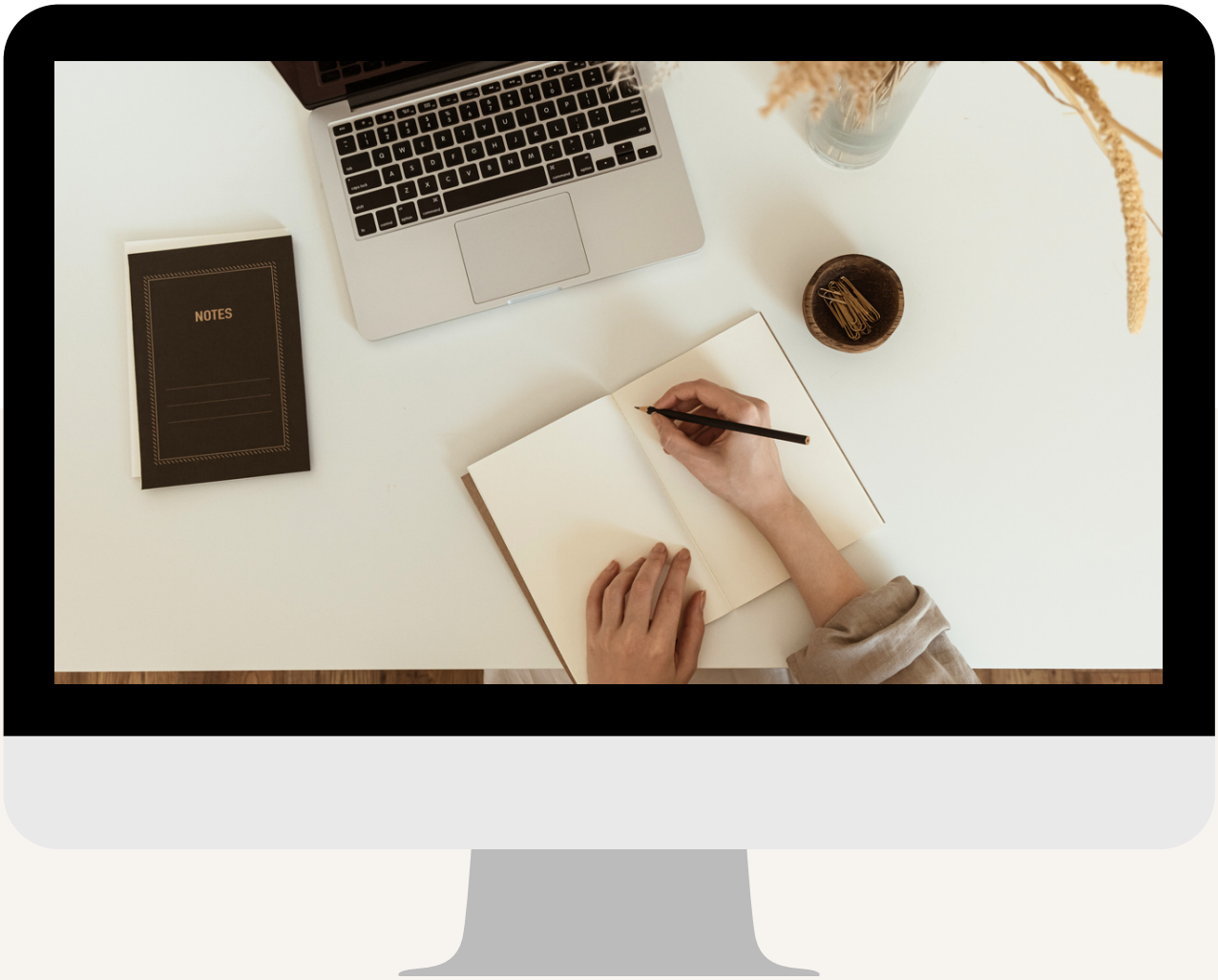
Современные веб-технологии стали неотъемлемой частью нашего быта. Мы ежедневно сталкиваемся с различными веб-приложениями и сайтами, базирующимися на сложных и мощных технологиях. Одним из основных языков программирования, лежащих в основе веб-технологий, является Java. Данный язык занимает одно из ведущих мест в мире IT и обладает обширным набором инструментов и фреймворков для разработки качественных веб-приложений.

Главным кирпичиком в структуре любого языка программирования является его синтаксис. Синтаксис – это правила, которые определяют структуру кода, его читаемость и корректность выполнения. Владение синтаксисом Java не только обеспечивает базовые навыки программирования на этом языке, но и является отправной точкой для погружения в более сложные и глубокие темы, такие как ООП, многопоточность, сетевое программирование и конечно же, фреймворки. Хорошее знание синтаксиса – это залог успешного изучения более продвинутых Java-технологий, которые используются в веб-разработке.

Важным аспектом разработки программного обеспечения является тестирование. Модульное тестирование позволяет разработчикам проверять отдельные компоненты или модули программы на корректность и соответствие заданным требованиям. Это обеспечивает высокое качество кода, уменьшает количество ошибок и позволяет быстро и эффективно модифицировать программу в будущем.

В ходе данной лабораторной работы мы рассмотрим основы синтаксиса языка Java и познакомимся с принципами модульного тестирования.

Эти знания станут фундаментом для дальнейшего изучения веб-технологий на платформе Java.



ХОД ВЫПОЛНЕНИЯ И ЗАЩИТЫ ЛАБОРАТОРНОЙ РАБОТЫ

Для выполнения лабораторной работы необходимо решить все задания, определенные в главе **Задания**

Для защиты лабораторной работы необходимо

- выполненные задания загрузить в гит-репозиторий
- сдать лабораторную работу в Moodle
- выполнить теоретический тест
- получить финальную оценку по лабораторной работе

ОБЩИЕ ТРЕБОВАНИЯ К КОДУ ЗАДАЧ, ВХОДЯЩИХ В ЛАБОРАТОРНУЮ РАБОТУ

- При написании кода обязательно используйте Java Code Convention. Именуйте переменные, методы, класс и прочее так, чтобы можно было понять назначение элемента. Не используйте сокращений, только если это не общепринятые сокращения.
- Не размещайте код всего приложения в одном методе (даже если задача вам кажется маленькой и “там же нечего писать”).
- Обязательно используйте пакеты.
- Не смешивайте в одном классе код, работающий с вводом-выводом данных, и логику, обрабатывающую эти данные (даже если вам кажется, что так быстрее). Создавайте разные типы классов: классы, объекты которых хранят данные, и классы, методы которых обрабатывают данные.
- Требование к наличию JUnit-тестов является обязательным.

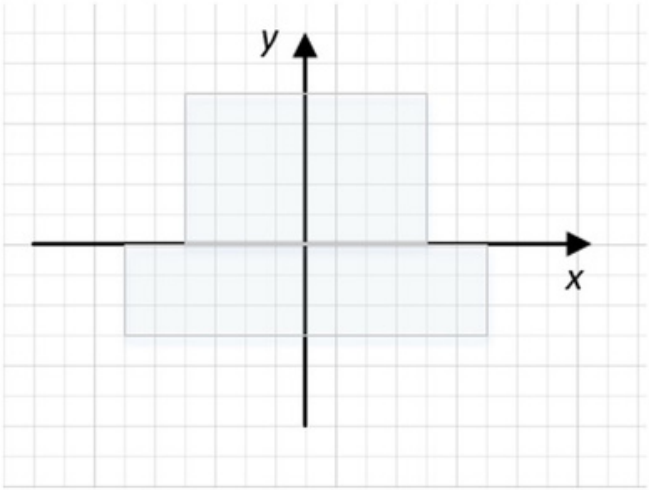
01. РЕШИТЕ ЗАДАЧУ

Вычислить значение выражения по формуле (все переменные принимают действительные значения). Для модульного тестирования приложения разработайте JUnit-тесты.

$$\frac{1+\sin^2(x+y)}{2+\left|x-\frac{2x}{1+x^2y^2}\right|}+x$$

02. РЕШИТЕ ЗАДАЧУ

Для данной области составить программу, которая печатает true, если точка с координатами (x, y) принадлежит закрашенной области, и false — в противном случае. Для модульного тестирования приложения разработайте JUnit-тесты.



03. РЕШИТЕ ЗАДАЧУ

Составить программу для вычисления значений функции F(x) на отрезке [a, b] с шагом h. Результат представить в виде таблицы, первый столбец которой – значения аргумента, второй - соответствующие значения функции. Для модульного тестирования приложения разработайте JUnit-тесты.

$$F(x) = tg(x)$$

04. РЕШИТЕ ЗАДАЧУ

Задан целочисленный массив размерности *N*. Определить, есть ли среди элементов массива простые числа. Если да, то вывести номера этих элементов. Для модульного тестирования приложения разработайте JUnit-тесты.

05. РЕШИТЕ ЗАДАЧУ

Дана целочисленная таблица *A[n]*. Найти наименьшее число *K* элементов, которые можно выкинуть из данной последовательности, так чтобы осталась возрастающая подпоследовательность. Для модульного тестирования приложения разработайте JUnit-тесты.

06. РЕШИТЕ ЗАДАЧУ

Даны действительные числа *a1*, *a2*, ..., *an*. Получить следующую квадратную матрицу порядка *n*. Для модульного тестирования приложения разработайте JUnit-тесты.

$$\begin{pmatrix} a_1 & a_2 & a_3 & \cdots & a_{n-2} & a_{n-1} & a_n \\ a_2 & a_3 & a_4 & \cdots & a_{n-1} & a_n & a_1 \\ a_3 & a_4 & a_5 & \cdots & a_n & a_1 & a_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ a_{n-1} & a_n & a_1 & \cdots & a_{n-4} & a_{n-3} & a_{n-2} \\ a_n & a_1 & a_2 & \cdots & a_{n-3} & a_{n-2} & a_{n-1} \end{pmatrix}$$

07. РЕШИТЕ ЗАДАЧУ

Дан массив n действительных чисел. Требуется упорядочить его по возрастанию. Делается это следующим образом: сравниваются два соседних элемента *ai* и *ai+1* . Если *ai <= ai+1*, то продвигаются на один элемент вперед. Если *ai > ai+1*, то производится перестановка и сдвигаются на один элемент назад. Составить алгоритм этой сортировки. Для модульного тестирования приложения разработайте JUnit-тесты.

08. РЕШИТЕ ЗАДАЧУ

Пусть даны две неубывающие последовательности действительных чисел *a1 <= a2 <= ... <=an* и *b1 <= b2 <= ... <= bm*. Требуется указать те места, на которые нужно вставлять элементы последовательности *b1 <= b2 <= ... <= bm* в первую последовательность так, чтобы новая последовательность оставалась возрастающей. Для модульного тестирования приложения разработайте JUnit-тесты.

09. РЕШИТЕ ЗАДАЧУ

Создать класс Мяч. Создать класс Корзина. Наполнить корзину мячиками. Определить вес мячиков в корзине и количество синих мячиков. Для модульного тестирования приложения создать JUnit-тесты. Составить алгоритм этой сортировки. Для модульного тестирования приложения разработайте JUnit-тесты.

10. РЕШИТЕ ЗАДАЧУ

Скомпилировать и запустить приложение, созданное при решении задачи 9 из командной строки.

11. РЕШИТЕ ЗАДАЧУ

Создать запускной jar-файл и запустить приложение, созданное при решении задачи 9-ть.

12. РЕШИТЕ ЗАДАЧУ

Переопределить методы *equals*(), *hashCode*() и *toString*(). Не пользуясь средствами автогенерации кода переопределить для класса Book методы *equals*(), *hashCode*() и *toString*().

```
public class Book {
        private String title;
        private String author;
        private int price;
        private static int edition;

}
```

13. РЕШИТЕ ЗАДАЧУ

Переопределить методы *equals*(), *hashCode*() и *toString*(). Не пользуясь средствами автогенерации кода переопределить для класса ProgrammerBook методы *equals*(), *hashCode*() и *toString*().

```
public class ProgrammerBook extends Book{
        private String language;
        private int level;

}
```

14. РЕШИТЕ ЗАДАЧУ

Переопределить метод *clone()*. Не пользуясь средствами автогенерации кода переопределить для класса Book из задачи 12 метод *clone()*. Напишите тесты JUnit, проверяющие корректность клонирования.

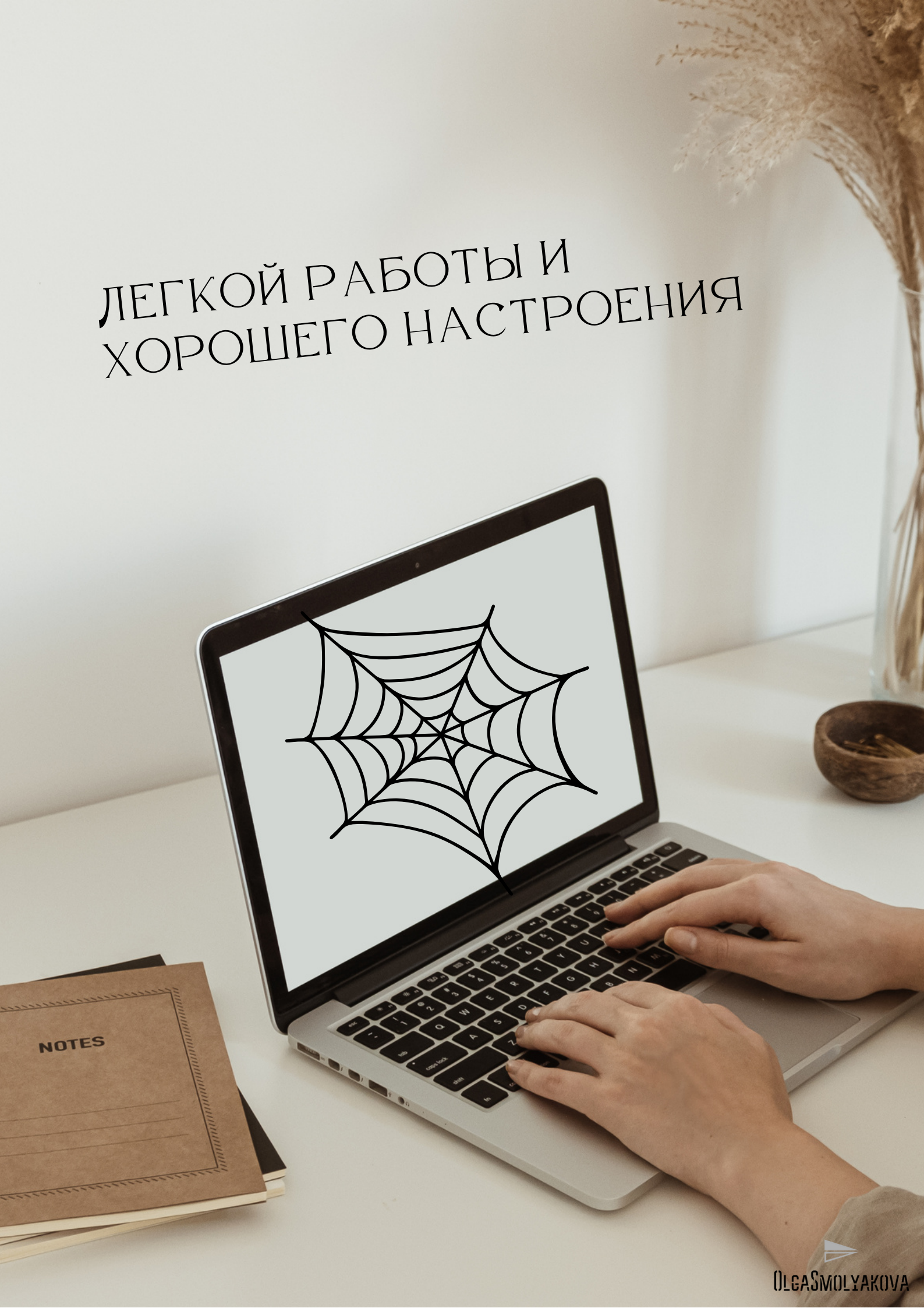
15. РЕШИТЕ ЗАДАЧУ

Реализовать интерфейс *Comparable*.
Добавьте в класс *Book* из задачи 12 поле *isbn*. Реализуйте в классе *Book* интерфейс *Comparable* так, чтобы книги приобрели сортировку по умолчанию согласно номеру *isbn*. Напишите тесты *JUnit*, проверяющие корректность сортировки.

16. РЕШИТЕ ЗАДАЧУ

Реализовать интерфейс *Comparator*.
Реализуйте для класса *Book* из задачи 12 компараторы, позволяющие сортировать книги по названию; по названию, а потом по автору; по автору, а потом по названию; по автору, названию и цене. Напишите тесты JUnit, проверяющие корректность сортировок.

ЛЕГКОЙ РАБОТЫ И ХОРОШЕГО НАСТРОЕНИЯ



OlgaSmolyakova