Веб-технологии

Совместный курс ИООО Эпам Системз и кафедры ПОИТ БГУИР

Лабораторная работа №1. Введение в Java

Общие требования к коду задач, входящих в лабораторную работу.

- При написании приложений обязательно используйте Java Code Convention.
- 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 данными, и логику (даже если вам кажется, что так быстрее). Создавайте разные типы классов: классы, объекты которых хранят данные, и классы, методы которых обрабатывают данные. Размещайте такие классы в разных пакетах.
- Требование к наличию Junit-тестов является обязательным.
- Именуйте переменные, методы, класс и прочее так, чтобы можно было понять назначение элемента. Не используйте сокращений, только если это не общепринятые сокращения.

Java Fundamentals

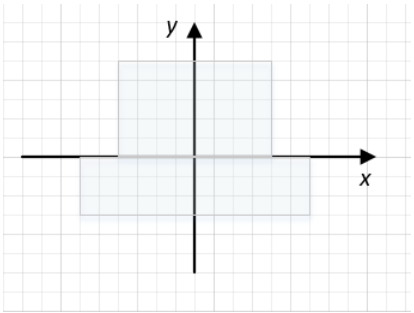
Задача 1. Решить задачу.

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

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

Задача 2. Решить 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 создать JUnit-тесты.



Задача 3. Решить 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 создать JUnit-тесты.

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

Задача 4. Решить задачу.

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

Задача 5. Решить 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 создать JUnit-тесты.

Задача 6. Решить задачу.

Даны действительные числа $a_1, a_2, \ldots, a_n$. Получить следующую квадратную матрицу порядка *n*. Для модульного тестирования приложения создать JUnit-тесты.

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

Задача 7. Решить задачу.

Сортировка Шелла. Дан массив n действительных чисел. Требуется упорядочить его по возрастанию. Делается это следующим образом: сравниваются два соседних элемента a_i и a_{i+1} . Если $a_i <= a_{i+1}$, то продвигаются на один элемент вперед. Если $a_i > a_{i+1}$, то производится перестановка и сдвигаются на один элемент назад. Составить алгоритм этой сортировки.

Задача 8. Решить задачу.

Пусть даны две неубывающие последовательности действительных чисел $a_1 <= a_2 <= \ldots <= a_n$ и $b_1 <= b_2 <= \ldots <= b_m$. Требуется указать те места, на которые нужно вставлять элементы последовательности $b_1 <= b_2 <= \ldots <= b_m$ в первую последовательность так, чтобы новая последовательность оставалась возрастающей.

Classes and Objects

Задача 9. Решить 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.

Задача 10. Создать и запустить приложение из командной строки

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

Задача 11.

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

Задача 12. 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;

}
```

Задача 13. 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;

}
```

Задача 14. Переопределить метод clone

Не пользуясь средствами автогенерации кода переопределить для класса Book из задачи 12 метод clone().

Задача 14. 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.

Задача 15. 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.