Лабораторная работа №4

Основы Java для автоматизации тестирования

Цель: Получить базовые сведения о языке Java и его возможностях для последующего использования полученных знаний в целях написания автоматизированных тестов.

План занятия:

1. Изучить теоретические сведения.
2. Выполнить практическое задание.

1. Теоретические сведения

Ознакомиться со сведениями представленные в справочнике для лабораторной №4

Перед началом работы необходимо будет установить следующее:

- Open JDK (https://openjdk.java.net)
- Java IDE

Рекомендуемая IDE - IntelliJ IDEA Community. Но можно воспользоваться любой другой на своё усмотрение.

2. Практические задания

Необходимо реализовать ряд небольших программ на языке Java в соответствии с условиями описанными в каждом задании.

Задание 1

Напишите класс **Vector**, описывающий вектор в трёхмерном пространстве, содержащий следующее:

- конструктор с параметрами в виде списка координат x, y, z;
- **double length()** - метод, вычисляющий длину вектора (корень можно посчитать с помощью Math.sqrt());

- **double scalarProduct(Vector vector)** - метод, вычисляющий скалярное произведение;
- **Vector crossProduct(Vector vector)** - метод, вычисляющий векторное произведение с другим вектором;
- **double cos(Vector vector)** - метод, вычисляющий угол между векторами (или косинус угла): косинус угла между векторами равен скалярному произведению векторов, деленному на произведение модулей (длин) векторов;
- **Vector add(Vector vector)** - методы для сложения векторов;
- **Vector add(Vector vector)** - методы для вычитания векторов;
- **static Vector[] generate(int n)** - статический метод, который принимает целое число n, и возвращает массив случайных векторов размером n.

Напишите небольшую демо программу демонстрирующую работу каждого из реализованных методов класса **Vector**

Задание 2

Реализуйте иерархию классов с оответствии с описанием ниже.

Класс Shape

Напишите класс **Shape** содержащий следующие поля и методы:

Поля:

- **String color** - поле задающее цвет формы в String формате (опционально можно использовать enum)

Методы:

- конструктор принимающий значения для всех полей класса и инициализирующий их;
- **String toString()** - переопределенный toString метод возвращающий строку вида:

```
Shape with <color_value> color
```

Класс Circle

Напишите класс **Circle** унаследованный от класса **Shape** и содержащий следующие поля и методы:

Поля:

- **double radius** - радиус круга

Методы:

- конструктор принимающий в качестве параметра **radius** и инициализирующий соответствующее поле класса. В случае если передано негативное значение радиуса полю **radius** должно быть выставлено значение 0.
- **double getArea()** - метод без параметров, вычисляющий площадь круга (можно использовать Math.PI)
- **String toString()** - переопределенный toString метод возвращающий строку вида:

```
A Circle with radius=<circle_radius> extending <parent_info>
```

где <parent_info> это значение возвращаемое toString() методом класса родителя.

Класс Rectangle

Класс **Rectangle** унаследованные от класса Shape и содержащий следующие поля и методы:

Поля:

- **double width** - ширина прямоугольника;
- **double height** - высота прямоугольника;

Методы:

- три конструктора:

    - **Rectangle()** - без параметров: должен выставлять width\length в 1.0;
    - **Rectangle(width, height)**;
    - **Rectangle(width, height, color)**
- **double getPerimeter()** - метод возвращающий периметр прямоугольника;
- **String toString()** - переопределенный toString метод возвращающий строку вида:

```
Rectangle with width=<rectangle_width> and height=<rectangle_height>
 extending <parent_info>
```

где <parent_info> это значение возвращаемое toString() методом класса родителя.

Класс Square

Напишите класс **Square** унаследованный от класса **Rectangle** и содержащий следующие поля и методы:

Поля:

Класс **Square** не содержит полей (используются поля унаследованные от класса **Rectangle**)

Методы:

- конструкторы:
    - **Square(double sideLength);**
    - **Square(side, color)**
- **String toString()** - переопределенный toString метод возвращающий строку вида:

```
Square with side=<side_length> extending <parent_info>
```

где <parent_info> это значение возвращаемое toString() методом класса родителя.

- **setLength()** и **setWidth()** - переопределенные методы класса **Rectangle** изменяющие ширину\высоту квадрата соответственно при вызове любого из этих методов для сохранения пропорций квадрата.

Напишите демо программу в которой создайте массив\список класса Shape и поместите в него как минимум один объект каждого класса **Circle**, **Rectangle** и **Square**. Реализуйте перебор элементов массива\списка и для каждого элемента выведите на экран:

- значение **toString** метода;
- цвет объекта;
- периметр или площадь (в зависимости от класса объекта)

Задание 3

Переделайте класс **Shape** из предыдущего задания сделав его абстрактным и добавив следующие абстрактные методы:

- **double getArea()** - абстрактный метод для возвращения площади фигуры;
- **double getPerimeter()** - абстрактный метод для возвращения периметра фигуры;
- **boolean isPointInside(double x, double y)** - абстрактный метод принимающий координаты точки (x, y) и возвращающий true\false в зависимости от того находится ли точка с такими координатами внутри заданной фигуры, полагая что центр каждой фигуры находится в точке с координатами (0, 0)

Обновите классы **Circle**, **Rectangle** и **Square** классы соответствующим образом (реализовав недостающие методы)

Напишите демо программу демонстрирующую работу обновленных классов и методов на примере объектов разного класса.

Задание 4

Напишите метод сравнивающие две коллекции типа **ArrayList**. Метод должен принимать в качестве параметров две коллекции типа **ArrayList**,

содержащие элементы одного из примитивных типов данных (например int) и возвращающий true\false в зависмости от того одинаковы ли элементы в этих колекциях. Порядок элементов в коллекциях значения не имеет. Важно лишь общее количество элементов и их значение. То есть для коллекций содержащие одинаковое количеством элементов с равными значениями, но имеющих разный порядок следования элементов метод должен возвращать true.

Напишите исполняемую программу, демонстрирующую работу метода на примере нескольких пар **ArrayList** коллекций.

Задание 5

Создайте свой собственный класс исключений (унаследовав его от **Exception**). Конструктор нового исключения должен принимать аргумент типа String и сохранть его внутри объекта исключения. Напишите метод, который распечатывает сохраненную строку. Создайте демо предложение демонстрирующее работу нового исключения (используя конструкцию **try-catch**)