Факультет программной инженерии и компьютерной техники

Тестирование программного обеспечения

Лабораторная работа №1

Вариант №33143

Преподаватель: Клименков С.В.

Выполнил: Тарасов А.С., P33112

Санкт-Петербург
2021

Задание

1. Для указанной функции провести модульное тестирование разложения функции в степенной ряд. Выбрать достаточное тестовое покрытие.
   2. Провести модульное тестирование указанного алгоритма. Для этого выбрать характерные точки внутри алгоритма, и для предложенных самостоятельно наборов исходных данных записать последовательность попадания в характерные точки. Сравнить последовательность попадания с эталонной.
   3. Сформировать доменную модель для заданного текста. Разработать тестовое покрытие для данной доменной модели

2. Функция arcsin(x)
3. Программный модуль для работы с красно-черным деревом (http://www.cs.usfca.edu/~galles/visualization/RedBlack.html)
4. Описание предметной области:

   И поскольку это далеко не самое естественное положение для кита, то у этого несчастного существа было очень мало времени на то, чтобы успеть свыкнуться с осознанием того, что оно кит, перед тем, как ему пришлось свыкнуться с осознанием того, что оно уже больше не кит.

Выполнение

Часть 1

Для выполнения первой части задания я воспользовался формулой разложения функции arcsin в ряд Тейлора:

$$\arcsin z = z + \left(\frac{1}{2}\right)\frac{z^3}{3} + \left(\frac{1\cdot 3}{2\cdot 4}\right)\frac{z^5}{5} + \left(\frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\right)\frac{z^7}{7} + \cdots =$$
$$= \sum_{n=0}^{\infty}\left(\frac{(2n)!}{2^{2n}(n!)^2}\right)\frac{z^{2n+1}}{(2n+1)}; \qquad |z| \le 1.$$

```
public static double arcsin(double x){
    if(x == 0) return 0.0;
    if(Double.isNaN(x)) return Double.NaN;
    if (x > 1 || x < -1) return Double.NaN;
     double res = 0;
    for(int n = 0; n < 20; n++){
        res = res + (factorial( a: 2*n)/(pow(2,2*n) * pow(factorial(n), 2))) * (pow(x,2*n+1))/(2*n+1);
    }
    return res;
}
```

Ошибка при разложении в ряд Тейлора увеличивается по мере удаления от аргумента, вокруг которого находится ряд. Это приводит к большим погрешностям на граничных значениях, поэтому два теста не проходят. Для более точного расчета используют формулу: $\arcsin(x) = \arctan(\frac{x}{\sqrt{1-x^2}})$.

Часть 2

В данной части мне пришлось реализовать модуль красно-черное дерево и методы работы с ним. Я ограничился вставкой элемента и поиском. Для этого реализованы некоторые классы:

Node.java – узел дерева

```
package SecondTask;

public class Node {

    private int key;
    private Node left, right, parent;
    private String color;


    public Node(){}

    public Node(String color){
        this.color = color;
    }
//getters & setters

}
```

RBTree.java – реализация самого дерева

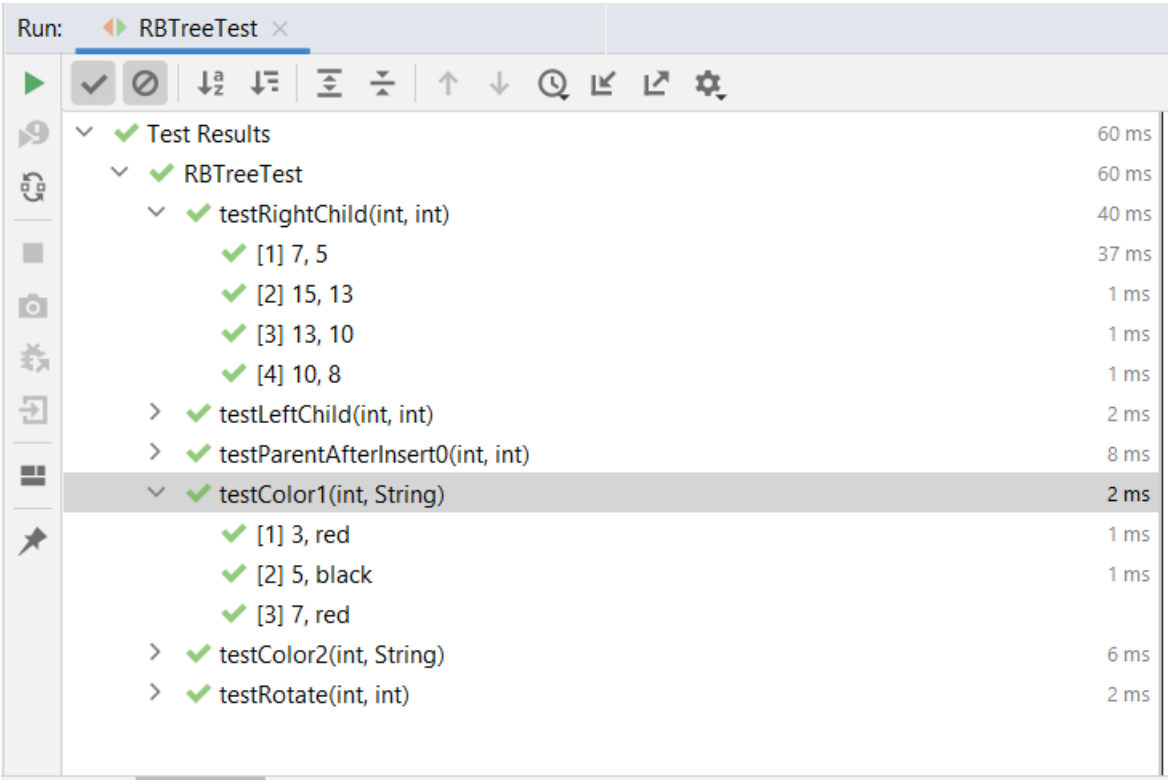
```
package SecondTask;



public class RBTree {
    public Node root = new Node("black");
    public Node nil = new Node("black");


    public void LeftRotate(Node x){
        Node y = x.getRight();
        x.setRight(y.getLeft());
        if(y.getLeft()!=this.nil){
            y.getLeft().setParent(x);
        }
        y.setParent(x.getParent());
        if(x.getParent() == this.nil){
            this.root = y;
        }
        else if(x == x.getParent().getLeft()){
            x.getParent().setLeft(y);
        }
        else {
            x.getParent().setRight(y);
        }
        y.setLeft(x);
        x.setParent(y);
    }

    public void RightRotate(Node x){
```

```
    Node y = x.getLeft();
    x.setLeft(y.getRight());
    if(y.getRight()!=this.nil){
        y.getRight().setParent(x);
    }
    y.setParent(x.getParent());
    if(x.getParent() == this.nil){
        this.root = y;
    }
    else if(x == x.getParent().getRight()){
        x.getParent().setRight(y);
    }
    else {
        x.getParent().setLeft(y);
    }
    y.setRight(x);
    x.setParent(y);
}


public void RBInsert(int key){
    Node t = new Node();
    t.setKey(key);
    Node y = this.nil;
    Node x = this.root;
    while(x != this.nil){
        y = x;
        if(t.getKey() < x.getKey()){
            x = x.getLeft();
        }
        else x = x.getRight();
    }
    t.setParent(y);
    if(y == this.nil){
        this.root = t;
    }
    else if(t.getKey()<y.getKey()){
        y.setLeft(t);
    }
    else{
        y.setRight(t);
    }
    t.setLeft(this.nil);
    t.setRight(this.nil);
    t.setColor("red");
    RBInsertFixUp(t);
}


public void RBInsertFixUp(Node z){
    if (z == this.root) {
        z.setColor("black");
        return;
    }
    while (z.getParent().getColor()=="red"){
        if(z.getParent() == z.getParent().getParent().getLeft()){ // если отец - левый
            Node y = z.getParent().getParent().getRight();//дядя

            if(y.getColor() == "red"){
                y.setColor("black");
                z.getParent().setColor("black");
                z.getParent().getParent().setColor("red");
                z = z.getParent().getParent();
```

```
            }else {
                if (z == z.getParent().getRight()) {
                    z = z.getParent();
                    LeftRotate(z);
                }
                z.getParent().setColor("black");
                z.getParent().getParent().setColor("red");
                RightRotate(z.getParent().getParent());
            }
        }
        else{     //если отец - правый
            Node y = z.getParent().getParent().getLeft(); //дядя
            if(y.getColor() == "red"){ //если дядя красный
                y.setColor("black");
                z.getParent().setColor("black");
                z.getParent().getParent().setColor("red");
                z = z.getParent().getParent();
            } else {    //если дяди нет
                if (z == z.getParent().getLeft()) {
                    z = z.getParent();
                    RightRotate(z);
                }
                z.getParent().setColor("black");
                z.getParent().getParent().setColor("red");
                LeftRotate(z.getParent().getParent());
            }
        }
    }
    this.root.setColor("black");
}



public Node Search(int k){
    Node x = this.root;
    while(x!=this.nil && k!=x.getKey()){
        if(k<x.getKey()){
            x = x.getLeft();
        }
        else {
            x = x.getRight();
        }
    }
    return x;
}

public void AllKey(Node x){
    if(x!=this.nil){
        AllKey(x.getLeft());
        System.out.println(x.getKey());
        AllKey(x.getRight());
    }
}
}
```

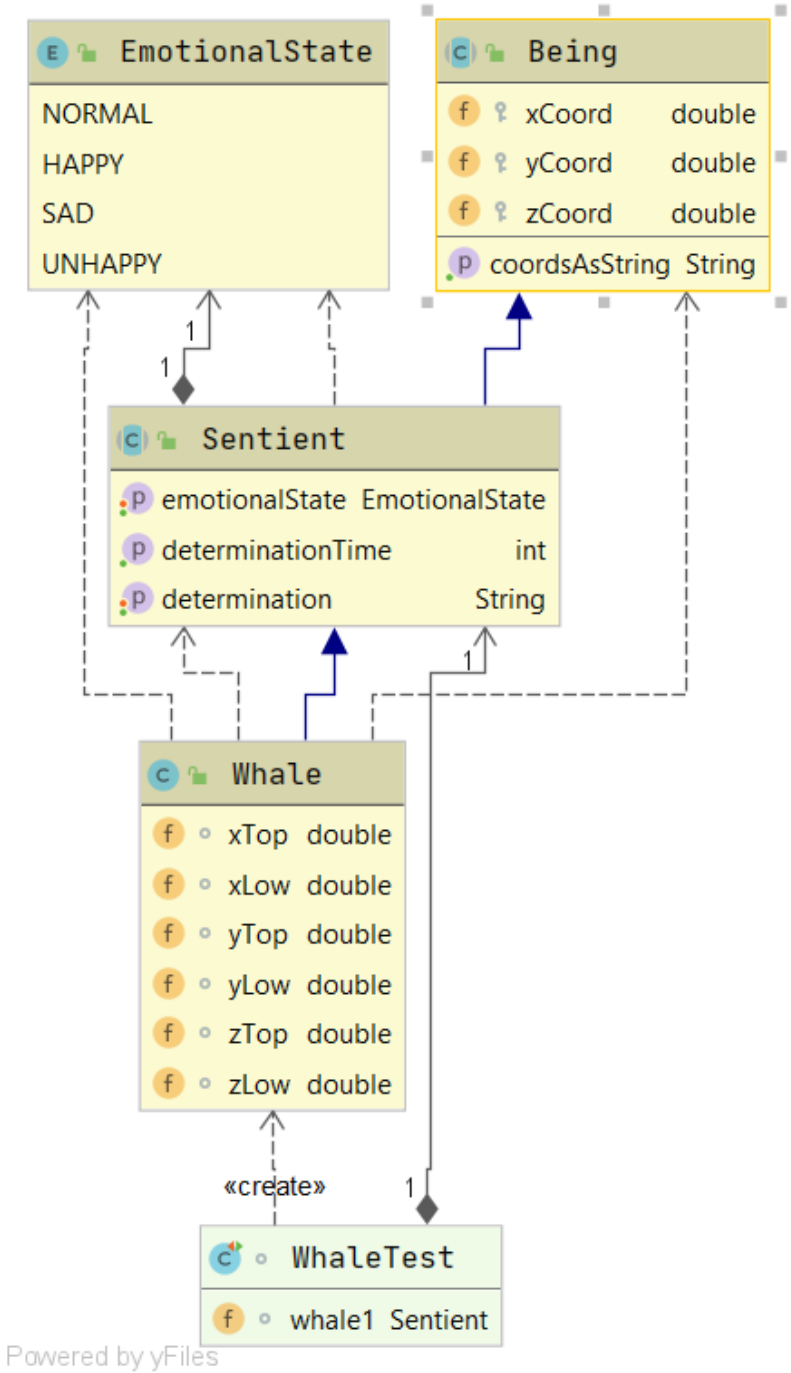
В ходе тестирования проверяются правильность вставки – по родителям, по цветам, по стороне ребенка (левый, правый)

Все тесты завершились успешно:



Часть 3

Реализована следующая модель:
«Кит» расширяет класс «Разумное», которое расширяет класс «Существо».



Код:

Being.java

```
package thirdTask;

public abstract class Being {
    protected double xCoord;
    protected double yCoord;
    protected double zCoord;


    protected void changePosition(double x, double y, double z){
        xCoord = x;
        yCoord = y;
        zCoord = z;
    }

//getters&setters
    public String getCoordsAsString(){
        return String.valueOf(getxCoord()) + " " + String.valueOf(getyCoord()) + " "  + String.valueOf(getzCoord());
    }
}
```

Sentient.java

```
package thirdTask;

public abstract class Sentient extends Being {
    EmotionalState emotionalState;
    protected String determination;

    int determinationTime;

    protected abstract String whoAmI();

    protected abstract void changeDetermination(String determination);

    public String getDetermination() {
        return determination;
    }

    protected void setDetermination(String determination) {
        this.determination = determination;
    }

    public EmotionalState getEmotionalState() {
        return emotionalState;
    }

    public void setEmotionalState(EmotionalState emotionalState) {
        this.emotionalState = emotionalState;
    }

    public int getDeterminationTime() {
        return determinationTime;
    }
}
```

Whale.java

```
package thirdTask;

public class Whale extends Sentient {

    double xTop = 10;
    double xLow = -10;
    double yTop = 50;
    double yLow = -50;
    double zTop = 50;
    double zLow = -50;

    Whale(){
        this.xCoord = 0;
        this.yCoord = 0;
        this.zCoord = 0;
        this.determination = "whale";
        this.determinationTime = 500;
        this.setEmotionalState(EmotionalState.NORMAL);
    }


    @Override
    public void changePosition(double x, double y, double z){
        xCoord = x;
        yCoord = y;
        zCoord = z;
        try {
            if (x > xTop || x < xLow || y > yTop || y < yLow || z > zTop || z < zLow) {
                determinationTime = 250;
                this.changeDetermination("not whale");
            }else {
                determinationTime = 500;
                this.changeDetermination("whale");
            }
            Thread.sleep(determinationTime);

        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    @Override
    protected void changeDetermination(String determination) {
        this.setDetermination(determination);
    }

    @Override
    public String whoAmI() {
        return this.toString();
    }


    @Override
    public String toString() {
        return "I am " + determination;
    }

}
```

Тестовый класс: WhaleTest.java

```
class WhaleTest {

    Sentient whale1;

    @BeforeEach
    public void initRBTree(){
        whale1 = new Whale();
    }


    @ParameterizedTest
    @CsvSource({
            "9, 48, 45",
            "-9, -53, 0",
            "-11, 48, 45",
            "12, 0, 0"
    })
    public void testChangePos(double x, double y, double z){
        whale1.changePosition(x, y, z); //ok
        assertEquals(String.valueOf(x) + " " + String.valueOf(y)+ " " + String.valueOf(z), whale1.getCoordsAsString()
);
    }

    @ParameterizedTest
    @ValueSource(strings = {  "whale", "not whale"})
    public void testChangeDetermination(String determination){
        whale1.changeDetermination(determination); //ok
        assertEquals(determination, whale1.getDetermination() );
    }


    @ParameterizedTest
    @CsvSource({
            "9, 48, 45, whale",
            "-9, -53, 0, not whale",
            "-11, 48, 45, not whale",
            "12, 0, 0, not whale"
    })
    public void testDet(double x, double y, double z, String det){
        whale1.changePosition(x, y, z); //ok
        assertEquals("I am "+det, whale1.whoAmI());
    }

    @ParameterizedTest
    @CsvSource({
            "9, 48, 45, 500",
            "-9, -53, 0, 250",
            "-3, 48, 45, 500",
            "12, 0, 0, 250"
    })
    public void testDetTime(double x, double y, double z, int detTime){
        whale1.changePosition(x, y, z); //ok
        assertEquals(detTime, whale1.getDeterminationTime());
    }
}
```

Вывод

Для выполнения данной работы потребовалось освоить основные навыки работы с JUnit, осознание того, что такое «юнит тесты», «метод черного ящика». Самая интересная и, в то же время, сложная часть данной работы заключается в умении придумать различные ситуации, модели поведения, и протестировать их на множестве разных наборов данных.