Исключения. Передача ссылок в конструктор. Повтор кода. Повторение Фабрик.

Практика 17 - Айрат Хасьянов

Golovolomka

На пустыре охотник столкнулся нос к носу с медведем. Оба испугались и побежали в разные стороны: охотник – на север, а медведь – на запад. Пробежав какое-то расстояние, охотник опомнился, направил ружьё точно на Юг, выстрелил и убил медведя.

Какого цвета был медведь?

Самостоятельная

- Создайте класс-фабрику для геометрических объектов. Фабричный метод должен возвращать окружность, квадрат или треугольник в зависимости от переданных параметров. Постройте геометрическую сцену, используя класс-фабрику.

Мои комментарии к домашним

Очень рациональная дробь

```
public class Task050 {
    public static void main(String[] args) {
        RationalFraction First = new RationalFraction();
        RationalFraction Second = new RationalFraction(5, 13);
        RationalFraction Third = First.add(Second);
        System.out.println(Third);

    }
}
```

Что можете сказать про этот код?

```
public  RationalFraction add(RationalFraction RF2) {
        RationalFraction r = new RationalFraction(numerator * RF2.getDenominator() +
RF2.getNumerator() * denominator,denominator * RF2.getDenominator());
        r.reduce();
        return r;
   }
```

toString()

- Как определен toString() в классе Object? Как это скажется на наших классах?
- Если в классе предопределен toString(), как выводить значения для объектов?

Комплексное число

```
public class ComplexNumber {
    private double a; //действительная часть
    private double b;  //мнимая часть
. . .
```

Присвоение матриц?

```
public void setMatrix(double[][] matrix) {
    this.matrix = matrix;
}
```

Страх и ужас!

```
public boolean equals(Vector2D v) {
    boolean f;
    if ((x == v.x) & (y == v.y)) {
        f = true;
    } else {
        f = false;
    }
    return f;
}
```

Отпустило…

```
public boolean equals(Vector2D v) {
    return (x == v.x) && (y == v.y);
}
```

Как убрать повтор кода?

```
private ComplexVector2D() {
    this.a = new ComplexNumber();
    this.b = new ComplexNumber();
}
```

Такой вариант мы знаем

```
private ComplexVector2D() {
    this(new ComplexNumber(), new ComplexNumber());
}
```

Attack of the Clones

```
public Matrix2x2(double matrix[][]) {

    double[][] Matrix = (double[][])matrix.clone();
}
```

Каков будет результат?

Вектор рациональных дробей

```
RationalVector2D(RationalFraction fraction1, RationalFraction fraction2) {
    this.fraction1.x = fraction1.x;
    this.fraction1.y = fraction1.y;
    this.fraction2.x = fraction2.x;
    this.fraction2.y = fraction2.y;
}
```

Это правильный код?

Комплексный вектор

```
ComplexVector2D(ComplexNumber complex1, ComplexNumber complex2) {
    this.complex1 = complex1;
    this.complex2 = complex2;
}
```

Почему так не стоит делать?

Что нужно исправить?

```
public RationalMatrix2x2 add(RationalMatrix2x2 RM2){
    RationalMatrix2x2 NewMatrix = new RationalMatrix2x2();
    for (int i=0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            Matrix[i][j].add(RM2.getMatrix(i,j));
        }
    }
    return NewMatrix;
}
```

Возвращаем ссылку на себя

```
public RationalMatrix2x2 add(RationalMatrix2x2 rationalMatrix){
   for (int i=0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            matrix[i][j].add(rationalMatrix.matrix[i][j]);
        }
    }
    return this;
}



class RationalMatrix2x2 {
    private RationalFraction [][] matrix = new RationalFraction[2][2];
```

Исключительный случай!

```
public Matrix2x2 add(Matrix2x2 matrix2x2){
    if(matrix.length != matrix2x2.matrix.length I matrix[0].length != matrix2x2.matrix[0].length){
        System.out.println("У матриц разные размеры, сложение невозможно");
    }
```

Это самый подходящий момент вспомнить ИСКЛЮЧЕНИЯ

Да, так тоже можно делать!

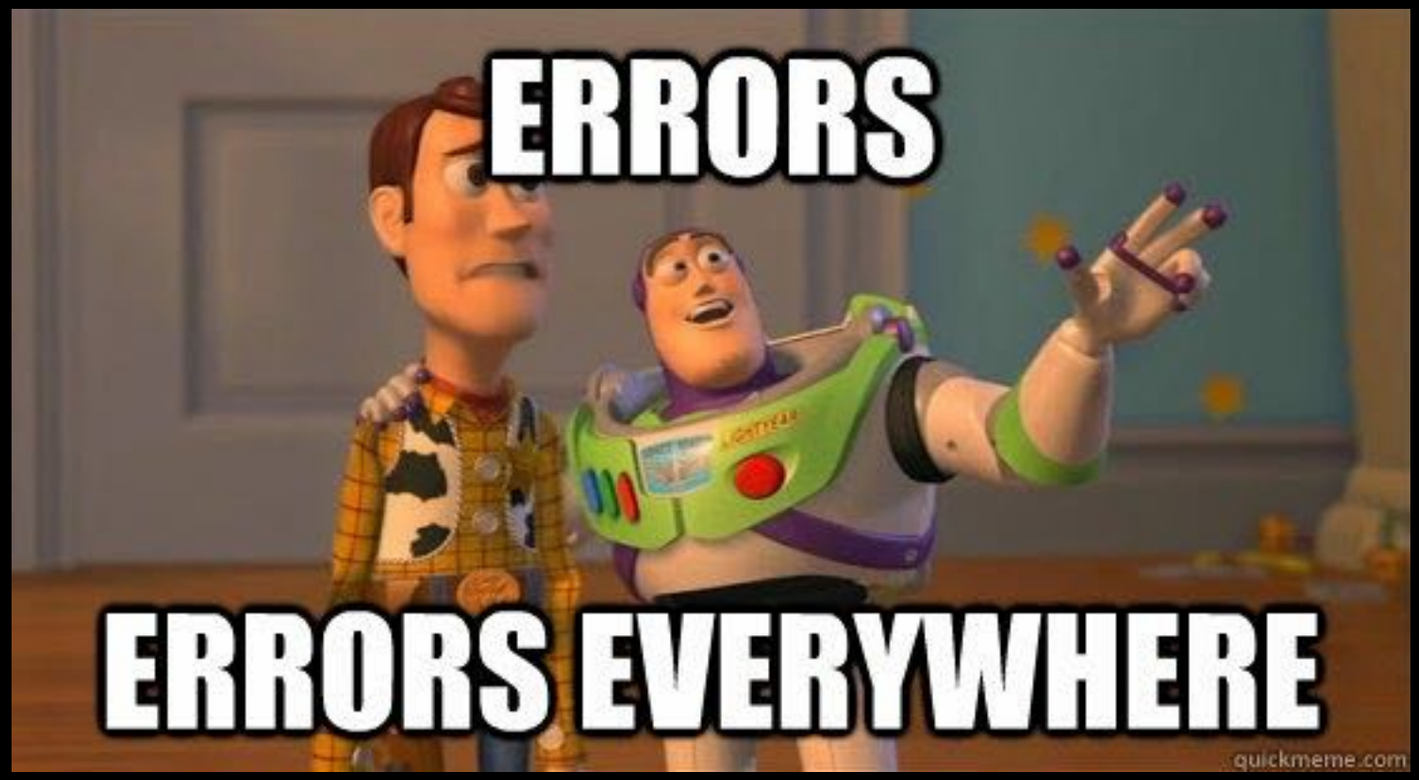
```
class RationalVector2D{
    private RationalFraction x = new RationalFraction();
    private RationalFraction y = new RationalFraction();

    public RationalVector2D(){}
```

Исключительный случай!

```
public Matrix2x2 add(Matrix2x2 matrix2x2) throws Exception {
    if(matrix.length != matrix2x2.matrix.length I matrix[0].length != matrix2x2.matrix[0].length){
        throw new Exception("У матриц разные размеры, сложение невозможно");
    }
```

Это самый подходящий момент вспомнить ИСКЛЮЧЕНИЯ


ERRORS
ERRORS EVERYWHERE
quickmeme.com

Исключения подтверждают правила

```
try {
    // блок кода, где отслеживаются ошибки
}
catch (тип_исключения_1 exceptionObject) {
    // обрабатываем ошибку
}
catch (тип_исключения_2 exceptionObject) {
    // обрабатываем ошибку
}
finally {
    // код, который нужно выполнить после завершения блока try
}
```

Исключения можем вызывать

throw экземпляр_Throwable

```
try {
   throw new NullPointerException("Матрицы не существует");
} catch (NullPointerException e) {
   System.out.println(e.getMessage());
}
```

О необработанных исключениях нужно предупреждать

```
тип имя_метода(список_параметров) throws список_исключений {
    // код внутри метода
}
```

Не такая уж и Самостоятельная

Создайте класс-фабрику для геометрических объектов. Фабричный метод должен возвращать окружность, квадрат или треугольник в зависимости от переданных параметров. Постройте геометрическую сцену, используя класс-фабрику.

Добавьте обработку исключений.

Класс Ticket

В классе предусмотреть методы, которые позволяют:

13. Проверить, является ли номер билета палиндромом (a

14. Проверить, является ли билет счастливым по-московски (abcdef => a+b+c = d+e+f)

15. Проверить, является ли билет счастливым по-питерски (abcdef=>a+c+e=b+d+f)

16. Проверить, является ли билет «почти счастливым» по московски [следующий или предыдущий к счастливому (abcdef => a+b+c = d+e+f)]

17. Проверить, является ли билет «почти счастливым» по питерски [следующий или предыдущий к счастливому (abcdef=>a+c+e=b+d+f)]