Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 3

по курсу «Языки и методы программирования»

«Полиморфизм на основе интерфейсов в языке Java»

Студент группы ИУ9-22Б Федуков А. А.

Преподаватель Посевин Д. П.

13 марта 2024 г.

Цель работы

Приобретение навыков реализации интерфейсов для обеспечения возможности полиморфной обработки объектов класса.

Задание

Во время выполнения лабораторной работы требуется разработать на языке Java один из классов, перечисленных в таблице. В классе должен быть реализован интерфейс Comparable<T> и переопределён метод toString. В методе main вспомогательного класса Test нужно продемонстрировать работоспособность разработанного класса путём сортировки массива его экземпляров.

Задание 1

Класс ферзей на шахматной доске, помнящих свою позицию, с порядком на основе количества ферзей, которых данный ферзь бьёт. (Потребуется дополнительный класс – шахматная доска.)

Реализация

Я описал класс Queen и Chessboard, исходя из условия задания, в файле Queen.java и Chessboard.java соответственно.
Сгенерировал его экземпляры и проверил работоспособность уже в Test.java

Код

Листинг 1 — Файл Queen.java

```
1  package task1;
2
3  public class Queen implements Comparable<Queen>{
4      private int x;
5      private int y;
6      private int numCouldEasilyCapture;
7
8      public void setNumCouldEasilyCapture(int n){this.
       numCouldEasilyCapture = n;}
9
10     public int getX() {
11         return x;
12     }
13
14     public int getY() {
15         return y;
16     }
17
18     public Queen(int x, int y){
19         this.x = x;
20         this.y = y;
21         this.numCouldEasilyCapture = 0;
22     }
23
24     @Override
25     public int compareTo(Queen obj) {
26         if (numCouldEasilyCapture==0 && obj.numCouldEasilyCapture==0)
       return 0;
27         else if (numCouldEasilyCapture == 0) return -1;
28         else if (obj.numCouldEasilyCapture == 0) return 1;
29         else return numCouldEasilyCapture - obj.numCouldEasilyCapture;
30     }
31
32     @Override
33     public String toString() {
34         return "task1.Queen{" +
35                 "x=" + x +
36                 ", y=" + y +
37                 ", numCouldEasilyCapture=" + numCouldEasilyCapture +
38                 '}';
39     }
40 }
```

Листинг 2 — Файл Chessboard.java

```
1  package task1;
2
3  import java.util.ArrayList;
4  import java.util.Collections;
5
6  public class ChessBoard {
7      private int numOfQueens;
8      private ArrayList<Queen> queens;
9
10     public void printQueens(){
11         if (numOfQueens == 0){
12             System.out.println("No queens though");
13         }
14         else {
15             for (int i = 0; i < numOfQueens; i++) {
16                 System.out.println(queens.get(i));
17             }
18         }
19     }
20     public void sortQueens(){
21         Collections.sort(this.queens);
22     }
23     private boolean CouldCapture(Queen q1, Queen q2){
24         return q1.getX() == q2.getX() | q1.getY() == q2.getY() | (Math.
       abs(q1.getX() - q2.getX()) == Math.abs(q2.getY() - q1.getY()));
25     }
26     private void MakeQueens(){
27         // Adding
28         for (int i = 0; i < numOfQueens; i++) {
29             queens.add(new Queen((int) (Math.random() * 8) + 1, (int) (
       Math.random() * 8) + 1));
30         }
31
32         // Balancing
33         for (int i = 0; i < numOfQueens; i++) {
34             int couldCapture = 0;
35             for (int j = 0; j < numOfQueens; j++) {
36                 if (i != j & CouldCapture(queens.get(i), queens.get(j)))
        {
37                     couldCapture += 1;
38                 }
39             }
40             queens.get(i).setNumCouldEasilyCapture(couldCapture);
41         }
42     }
43     public ChessBoard(int numOfQueens){
44         this.numOfQueens = numOfQueens;
45         this.queens = new ArrayList<Queen>();
46         MakeQueens();
47     }
48 }
```

Листинг 3 — Файл Test.java

```
1  package task1;
2
3  public class Test {
4      public static void main(String[] args) {
5
6          ChessBoard CB = new ChessBoard(6);
7          CB.printQueens();
8          CB.sortQueens();
9          System.out.println("Queens sorted btw");
10         CB.printQueens();
11
12     }
13 }
```

Вывод программы

Программа создала 6 ферзей и отсортировала их по количеству других ферзей, которых бьет данный ферзь. Причем в данном случае для разнообразия был использован ArrayList, а в задании 2 просто Array.

Листинг 4 — Вывод программы

```
1  task1.Queen{x=1, y=2, numCouldEasilyCapture=0}
2  task1.Queen{x=3, y=6, numCouldEasilyCapture=2}
3  task1.Queen{x=3, y=3, numCouldEasilyCapture=1}
4  task1.Queen{x=6, y=8, numCouldEasilyCapture=0}
5  task1.Queen{x=2, y=5, numCouldEasilyCapture=1}
6  task1.Queen{x=7, y=1, numCouldEasilyCapture=0}
7  Queens sorted btw
8  task1.Queen{x=1, y=2, numCouldEasilyCapture=0}
9  task1.Queen{x=6, y=8, numCouldEasilyCapture=0}
10 task1.Queen{x=7, y=1, numCouldEasilyCapture=0}
11 task1.Queen{x=3, y=3, numCouldEasilyCapture=1}
12 task1.Queen{x=2, y=5, numCouldEasilyCapture=1}
13 task1.Queen{x=3, y=6, numCouldEasilyCapture=2}
14
15 Process finished with exit code 0
```

Задание 2

Класс целых чисел с порядком на основе количества простых делителей.

Реализация

Я создал класс MyInt, заточенный под счет простых делителей.
Сгенирировал его экземпляры и проверил работоспособность уже в Test.java

Код

Листинг 5: Файл MyInt.java

```java
package task2;

public class MyInt implements Comparable<MyInt>{
    private int num;
    private int numPriveDivs;
    private int countNumPrimeDivs(){

        int d = 2;
        int count = 1;
        int n = num;

        while (d * d <= n){
            if (n % d == 0){
                n = n / d;
                count += 1;
            }
            else {
                d += 1;
            }
        }
        if (n > 1){
            count += 1;
        }

        return count;
    }
    public MyInt(int num){
        this.num = num;
        this.numPriveDivs = countNumPrimeDivs();

    }
```

```
32      @Override
33      public int compareTo(MyInt obj) {
34          if (numPriveDivs==0 && obj.numPriveDivs==0) return 0;
35          else if (numPriveDivs == 0) return -1;
36          else if (obj.numPriveDivs == 0) return 1;
37          else return numPriveDivs - obj.numPriveDivs;
38      }
39
40      @Override
41      public String toString() {
42          return "MyInt{" +
43                  "num=" + num +
44                  ", numPriveDivs=" + numPriveDivs +
45                  '}';
46      }
47  }
```

Листинг 6 — Файл Test.java

```
1   package task2;
2
3   import java.util.Arrays;
4
5   public class Test {
6       public static void main(String[] args) {
7           // Разница между num и this.num???
8           // Что значит @Override
9           // Когда var public, а когда getVar и private
10          int n = 10;
11
12          MyInt[] myIntArrayList = new MyInt[n];
13
14          for (int i = 0; i < n; i++) {
15              myIntArrayList[i] = (new MyInt((int) (Math.random() * 100)))
        ;
16          }
17
18          System.out.println("List before sort");
19          for (int i = 0; i < n; i++) {
20              System.out.println(myIntArrayList[i]);
21          }
22
23          Arrays.sort(myIntArrayList);
24          System.out.println("\nList after sort");
25          for (int i = 0; i < n; i++) {
26              System.out.println(myIntArrayList[i]);
27          }
28
29      }
30  }
```

Вывод программы

Программа создала 10 объектов и протестировала метод сортировки по количеству простых делителей.

Листинг 7 — Вывод программы

```
List before sort
MyInt{num=89, numPriveDivs=2}
MyInt{num=14, numPriveDivs=3}
MyInt{num=11, numPriveDivs=2}
MyInt{num=15, numPriveDivs=3}
MyInt{num=36, numPriveDivs=5}
MyInt{num=96, numPriveDivs=7}
MyInt{num=52, numPriveDivs=4}
MyInt{num=78, numPriveDivs=4}
MyInt{num=58, numPriveDivs=3}
MyInt{num=15, numPriveDivs=3}

List after sort
MyInt{num=89, numPriveDivs=2}
MyInt{num=11, numPriveDivs=2}
MyInt{num=14, numPriveDivs=3}
MyInt{num=15, numPriveDivs=3}
MyInt{num=58, numPriveDivs=3}
MyInt{num=15, numPriveDivs=3}
MyInt{num=52, numPriveDivs=4}
MyInt{num=78, numPriveDivs=4}
MyInt{num=36, numPriveDivs=5}
MyInt{num=96, numPriveDivs=7}

Process finished with exit code 0
```

Вывод

Я продолжил освоение программирования на Java, в этот раз я научился создавать объекты с интерфейсами, которые позволяли работать с этими объектами полимофно, то есть применять к ним некоторые методы и тд вне зависимости от их типа.