Report on s1709906 's revision attempt at Inf1OP Programming Exam (sitting 2)

Generated by Automarker on Mon Apr 23 18:40:40 BST 2018

Question 1

Part 1all

Compiling UG5Student.java with the basic tests given to students in the exam succeeded.

Passed all 3 basic tests.

Compiling the submitted UG5Student.java with the test file worked fine.

Passed all 0 tests.

Marks for this part: 50 / 50

Submitted UG5Student.java

```
1  class UG5Student extends Student {
2
3   private int extraLevel9s;
4
5   public UG5Student(String name, String uun, int extraLevel9s
    ) {
6    super(name, uun, 5);
7    this.extraLevel9s = extraLevel9s;
8   }
9
10  public UG5Student() {
11   this("not␣set", "not␣set", 0);
12  }
13
14  public int level9s() {
15
16   int level9courses = 0;
17
18   for (Course c : getCourses()) {
```

```
19      if (c.getLevel() == 9) {
20       level9courses += 1;
21      }
22     }
23
24     return level9courses;
25    }
26
27    @Override
28    public boolean addCourse(Course course) {
29     int max = extraLevel9s + 1;
30     int current = level9s();
31
32     if (current >= max) {
33      return false;
34     }
35
36     super.addCourse(course);
37     return true;
38    }
39
40    @Override
41    public String toString() {
42     String s = super.toString();
43
44     if (level9s() > 1) {
45      s = s + "Taking␣extra␣level␣9s.␣Permission␣for␣" +
      extraLevel9s + "␣extra.";
46     }
47
48     for (Course c : getCourses()) {
49      s = s + "\n" + c.getName();
50     }
51
52     return s;
53    }
54
55   }
```

Marks for Question 1: 50/50

Question 2

Part 2all

Compiling Fermat.java with the basic tests given to students in the exam succeeded.

Passed all 4 basic tests.

Compiling the submitted Fermat.java with the test file worked fine.

There were 1 failures out of the 18 tests.

The tests that failed were:

testThreadings0redgreen

Problem	Key evidence	Effect on mark
2a) base case behaviour with n=0 wrong	0 test failed	-1 Total: -1

Marks for this part: 49 / 50

Submitted Fermat.java

```
1  import java.util.ArrayList;
2  import java.util.Arrays;
3  import java.util.Collections;
4  import java.util.HashSet;
5  import java.util.Set;
6
7  public class Fermat {
8
9   public static Set<ArrayList<String>> threadings(int n, Set<
   String> colours) {
10   // your code must not alter the initial set, so we need to
    create a new set
11   Set<ArrayList<String>> necklaces = new HashSet<ArrayList<
   String>>();
12
13   if ( n < 1 ) {
14    return new HashSet<>();
15   }
16
17   //ArrayList<String> necklace = new ArrayList<String>();
18
19   for (String firstColour : colours) {
20    if ( n == 1 ) {
21     ArrayList<String> necklace = new ArrayList<String>();
22     necklace.add(firstColour);
23     necklaces.add(necklace);
24    }
25
26    for (ArrayList<String> otherColours : threadings(n-1,
   colours)) {
27     ArrayList<String> necklace = new ArrayList<String>();
28     necklace.add(firstColour);
```

```
29      necklace.addAll(otherColours);
30      necklaces.add(necklace);
31
32     }
33
34    }
35
36    return necklaces;
37   }
38
39   public static boolean isEquiv(ArrayList<String> necklace1,
     ArrayList<String> necklace2) {
40    // rotate the necklace
41
42    boolean flag = false;
43    boolean rotateYes = false;
44    boolean reverseYes = false;
45    ArrayList<String> rotated = new ArrayList<String>(
     necklace1);
46
47    for (int i = 0; i < necklace1.size(); i++) {
48     if (rotated.equals(necklace2)) {
49      rotateYes = true;
50     }
51     Collections.rotate(rotated, i); // after rotating
52     if ( rotated.equals(necklace2)) {
53      rotateYes = true;
54      break;
55     } else {
56      Collections.reverse(rotated); // after reversing
57      if (rotated.equals(necklace2)) {
58       reverseYes = true;
59       break;
60      }
61     }
62    }
63
64    if (rotateYes || reverseYes) {
65     flag = true;
66    }
67
68    return flag;
69   }
70
71   public static void analyse(Set<ArrayList<String>> s) {
72
73   Set<ArrayList<String>> singletons = new HashSet<ArrayList<
     String>>(s);
74   boolean equivalent = false;
75   ArrayList<ArrayList<String>> necklaces = new ArrayList<
```

```
ArrayList<String>>(s);
 for (int i = 0; i < s.size(); i++) {
  for (int j = 0; j < s.size(); j++) {
   if ( i != j ) {
    equivalent = isEquiv(necklaces.get(i), necklaces.get(j)
);
    if (equivalent) {
     singletons.remove(necklaces.get(i));
    }
   }
  }
 }

 for (ArrayList<String> singleton : singletons) {
  System.out.println(Arrays.toString(singleton.toArray()) +
 "␣is␣a␣singleton");
 }
}

public static void main(String[] args) {
 int n = Integer.parseInt(args[0]);
 Set<String> colours = new HashSet<String>();

 for (int i = 1; i < args.length; i++) {
  colours.add(args[i]);
 }

 Set<ArrayList<String>> threaded = threadings(n, colours);

 analyse(threaded);

}

// public static void main(String[] args) {
//  ArrayList<String> n1 = new ArrayList<String>();
//  n1.add("red");
//  n1.add("green");
//  n1.add("blue");
//  n1.add("pink");
//
//  ArrayList<String> n2 = new ArrayList<String>();
//  n2.add("red");
//  n2.add("pink");
//  n2.add("blue");
//  n2.add("green");
//
//  // System.out.println(isEquiv(n1,n2));
//
//  Set<ArrayList<String>> s = new HashSet<ArrayList<String
 >>();
```

```
122 //  ArrayList<String> n3 = new ArrayList<String>();
123 //  n3.add("red");
124 //  n3.add("red");
125 //
126 //  ArrayList<String> n4 = new ArrayList<String>();
127 //  n4.add("red");
128 //  n4.add("green");
129 //
130 //  ArrayList<String> n5 = new ArrayList<String>();
131 //  n5.add("green");
132 //  n5.add("red");
133 //
134 //  s.add(n3);
135 //  s.add(n4);
136 //  s.add(n5);
137 //
138 //  analyse(s);
139 //
140 //
141 // }
142 }
```

Marks for Question 2: 49/50

Total marks: 99 / 100