University of Glasgow

Saturday 12 December 2020, 13:00
(Timed online assessment – Indicative duration 1 hour)

DEGREES of MSci, MEng, BEng, BSc, MA and MA (Social Sciences)

JAVA PROGRAMMING 2 (COMPSCI 2001)

Answer all 3 questions

This examination paper is worth a total of 50 marks.

Important note: throughout this exam, whenever you are asked to write Java code, do not worry about whether your code compiles. The markers will never test any submitted code from this exam.

1. This question concerns Java class and method design. (20 marks total)

First, read the following description of a sports league.

> Your task is to model league of teams that play a sport called Javaball. Each **team record** has two properties:
>
> - A team name (a string)
> - The total number of **points** earned from their games (an integer)
>
> Every game in the league involves two teams, and the winner is the team that scores more goals. If both teams score the same goals, the game is a draw.
>
> After a game is completed, the **points** for each team are updated as follows:
>
> - If the game is a draw, both teams get **1 point** added to their total
> - Otherwise, the winning team gets **3 points** added to their total, and the losing team's points total is unchanged.

(a) Write a full class definition for **TeamRecord** following the specification above. Be sure to use appropriate types and access modifiers for the fields. Include a constructor that initialises all fields: the initial values for points should be zero. Also include a getter method for all fields. [5]

Solution:

```
// 1 mark for class signature
public class TeamRecord {
    // 0.5 marks for good types
    // 0.5 marks for access modifier
    private String name;
    private int points;

    // 0.5 marks for constructor signature
    public TeamRecord (String name) {
        // 0.5 marks for this assignment
        this.name = name;

        // 0.5 marks for this assignment
        this.points = 0;
    }

    // 1 mark for all getters
    // 0.5 marks for correct access modifiers
    public String getName() {
```

```
        return this.name;
    }

    public int getPoints() {
        return this.points;
    }
}
```

(b) Write a method for the **TeamRecord** class to update the team record with the result of a new match. The method should take two integer parameters, representing the goals scored by the current team and the opposing team, respectively, and should update the team's total points based on the description above. [3]

Solution:

```
// 1 mark for header
public void updateRecord (int myGoals, int otherGoals) {
    // 1 mark for "win" case
    if (myGoals > otherGoals) {
        this.points += 3;
    // 1 mark for "draw" case
    } else if (myGoals == otherGoals) {
        this.points += 1;
    }
}
```

(c) There are 12 teams in the Javaball league. Identify an appropriate data type to use for representing the records of all 12 teams in the league. Show a line of code declaring and initialising a variable of this type to represent the collection of teams. [2]

Solution: One mark for plausible data type; one mark for initialising that data type correctly

```
// An array is possible -- must be initialised correctly
Team[] premierTeams = new Team[12];

// Could also use either of the Collections classes below
// Don't need to include 12 in the constructor
// but it's not wrong if they do
List<Team> premierTeams = new ArrayList<>();
Set<Team> premierTeams = new HashSet<>();
```

(d) Write a `compareTo` method for the `TeamRecord` class that compares two `TeamRecord`

objects as follows:

- If one team has more points than the other, then the team with more points comes first.
- If two teams have the same points, then the team that comes earlier in the alphabet comes first.

[6]

Solution:

```
// 1.5 marks for header (access modifier, return type, param)
public int compareTo (TeamRecord other) {
    // 1 marks for including points comparison
    // .5 marks for getting it the right way around
    int result = Integer.compare (other.points, this.points);

    // 1 mark for getting this condition correct
    if (result == 0) {
        // 1 mark for including name comparison
        // .5 marks for right way around
        result = String.compare (this.name, other.name);
    }

    // 0.5 marks for remembering return statement
    return result;
}
```

(e) How would you need to modify the class header for `TeamRecord` to make full use of the implemented `compareTo` method? [2]

Solution:

```
// 1 mark for implementing Comparable
// 1 mark for including generic parameter
public class TeamRecord implements Comparable<TeamRecord> {
    // class body
}
```

(f) Give a line of code that shows how you would sort the collection of `TeamRecord` objects identified in part (c), using the sort order defined in parts (e) and (f). [2]

Solution:

```
// If an array is used above
Arrays.sort (premierTeams);
```

```
// If a Collection class is used
Collections.sort (premierTeams);

// Could also do something with streams,
// or a TreeSet/SortedSet, etc
```

2. This question asks you to read and understand Java code. (20 marks total)

(a) Study the following Java class and answer the questions that follow:

```
1  public class ExamQ2 {
2          private int a;
3          public ExamQ2(int a) {
4                  this.a = a;
5          }
6          private int getA() {
7                  return this.a;
8          }
9          private void setA(int a) {
10                 this.a = a;
11         }
12         public static void add3(int a) {
13                 a += 3;
14         }
15         public static void add3Obj(ExamQ2 e) {
16                 e.setA(3 + e.getA());
17         }
18         public static void main(String[] args) {
19                 int a = 5;
20                 add3(a);
21                 System.out.println(a);
22
23                 ExamQ2 ee = new ExamQ2(a);
24                 add3Obj(ee);
25                 System.out.println(ee.getA());
26         }
27 }
```

(i) Line 9 contains a method signature. Copy the signature and label each of its components, describing the meaning of each. [4]

Solution:

- private: access modifier, indicates that the method can only be called from within the current class
- void: return type, indicates that the method does not return anything
- setA: method name, how to call it
- int b: parameter name and type

One mark for each component

(ii) The `println` statement at Line 21 prints **"5"**. Explain clearly why this is the case, referring to the behaviour of the statements on lines 19 and 20 and the `add3` method. [3]

Solution:

- Line 19: a is initialised to 5
- Line 20: the static method is called with a as its parameter
- The method updates the local value of the parameter, but does not change the value outside the method

Or any similar explanation that demonstrates understanding

(iii) The `println` statement at Line 25 prints **"8"**. Explain clearly why this is the case, referring to the behaviour of the statements at lines 23 and 24 and the `add3Obj` method. [3]

Solution:

- Line 23: A new `ExamQ2` object `ee` is created with `a` as its parameter, which it stores in the field
- Line 24: The static `add3Obj` method is called with `ee` as its parameter
- Inside `add3Obj`, the field value is accessed with `getA` and then updated with `setA`, which changes the value for the println statement

Or any similar explanation that demonstrates understanding

(b) Study the following Java code and then answer the questions that follow. Note that the method `Character.isWhitespace(c)` returns **`true`** if `c` is a whitespace character (i.e., a space, tab, or similar) and **`false`** if it is not.

```
1  public class Trims {
2     /* Removes whitespace at the beginning of a string */
3     public static String leftTrim (String s) {
4        int i = 0;
5        while (i < s.length() && Character.isWhitespace(s.charAt(i))) {
6           i++;
7        }
8        return s.substring(i);
9     }
10
11    /* Removes whitespace at the end of a string */
12    public static String rightTrim (String s) {
13       int i = s.length() - 1;
14       while (i > 0 && Character.isWhitespace(s.charAt(i))) {
15         i--;
16       }
17       return s.substring(0, i + 1);
18    }
19
20    public static void main (String[] args) {
21         String s = "  Trim me     ";
22         System.out.println(s);
23         System.out.println(leftTrim(s));
24         System.out.println(rightTrim(s));
25    }
26 }
```

(i) Explain why the methods `rightTrim()` and `leftTrim()` need to be declared **`static`**. What changes would be necessary in the `main` method if that modifier was removed? [4]

> **Solution:**
> They need to be static because they are called directly from the static `main` method. (2 marks)
> If they were not static, there would need to be an instance to call them on, so `main` would need to construct objects before calling the methods. (2 marks)

(ii) Would it make a difference to the operation of the `main` method if `leftTrim()` and `rightTrim()` were declared as **`private`** instead of **`public`**? Why or why not? [2]

Solution: No, it wouldn't make any difference – private methods are only visible within the declaring class, but here they are called from the `main` method so there is no need for them to be public.

(iii) Write a single line of code that you could add to the above `main` method to print out a string that is trimmed both on the left and on the right. [2]

Solution:

```
// rightTrim(leftTrim(s)) would also work
System.out.println(leftTrim(rightTrim(s)));
```

(iv) Briefly explain why the programmer might have chosen to use **`while`** loops rather than **`for`** loops in `leftTrim()` and `rightTrim()`. [2]

Solution: The main point would be because they need to use the value of `i` after the end of the loop. Other valid explanations also acceptable as long as they show understanding of the distinction between the two types of loop.

3. This question asks you to understand and discuss Java programming concepts (10 marks total)

(a) Explain, with reference to the concept of polymorphism, why it is necessary to override `hashCode()` whenever you override `equals()` in a class. [4]

Solution: Points to mention:

- Many built-in Java classes (e.g., HashMap/HashSet) use `hashCode()` and `equals()` together to determine how to store a value.
- In particular, they assume that if two values are equal according to `equals()` then they have the same hash code.
- When any code calls built-in methods such as `equals()` and `hashCode()` (or in fact any methods), they will get the most specific version of that method, regardless of the declared type (*polymorphism*)
- This means that if you override `equals()` but not `hashCode()`, the calling code will use the overridden version of `equals()` but the default version of `hashCode()`, leading to potential weirdness.

One mark for mentioning each point (or similar ones)

(b) What would change in Java if all exceptions were treated as **unchecked** exceptions? Give one potential advantage and one potential disadvantage of such a change. [3]

Solution: If all exceptions were unchecked, then a programmer would never need to use try/catch unless they chose to, and any exception could crash a program if uncaught. (1 mark)

Advantage: easier to write code if you can ignore all exceptions you don't want to have to deal with (1 mark)

Disadvantage: makes it easier to ignore error situations that really should be dealt with, code will crash more often (e.g., file not found) (1 mark)

Marks also for similar plausible answers

(c) What would change in Java if all exceptions were treated as **checked** exceptions? Give one potential advantage and one potential disadvantage of such a change. [3]

Solution: If all exceptions were checked, then a programmer would always need to use try/catch on every possible error situation (1 mark)

Advantage: all error situations would be caught and a program could never crash (1 mark)

Disadvantage: you'd need to have try/catch on basically every line of code (in case of NullPointerException) (1 mark)

Marks also for similar plausible answers