The Open University

M255/SSEP

Mathematics, Computing and Technology: Level 2
M255 Object-oriented programming with Java

M255 Specimen Exam Solutions

Part 1

1. AD
2. CD
3. E
4. BE
5. DE
6. DE
7. AC
8. BF
9. AD
10. AE
11. CF
12. BE
13. BC
14. BC
15. AD
16. AD
17. CE
18. B
19. AD
20. CD

Part 2

Question 21 [10 marks]

(i) Two objects exist. The first two statements each involve the creation of a new Frog object but the third statement simply makes `hoverFrog3` reference the same object as `hoverFrog1`. [2 marks]

(ii)
```
/**
 * Sets the position of the receiver to be the same as
 * the position of the argument.
 */
public void samePositionAs(Frog aFrog)
{
   this.setPosition(aFrog.getPosition());
}
```

[4 marks]

(iii) The method `samePositionAs(Frog)` is first looked for in the class of the receiver (`HoverFrog`) but is it not found. A method with this signature is now looked for in the superclass, `Frog`, where it is found. It is executed, with the actual argument (the `Frog` object referenced by `frog1`) being assigned to the formal argument (and `this` being replaced by the receiver). The result is that the position instance variable of the receiver, `hoverFrog1` is set to the same position as the argument object, `frog1`. [4 marks]

Question 22 [15 marks]

(i)
```
public interface Danceable
{
   public void pirouette(int anInteger);
   public void prepareToDance();
}
```

[3 marks]

(ii)
```
public class DanceableHoverFrog() extends HoverFrog implements
Danceable
{
   public DanceableHoverFrog()
   {
      super();
   }

public void pirouette(int anInteger)
{
   for (int i = 0;  i < anInteger; i++)
   {
      this.up();
      this.left();
      this.down();
      this.right();
   }
}

public void prepareToDance()
{
   while (this.getPosition() > 5)
   {
      this.setHeight(3); this.left(); this.downBy(3);
   }
  while (this.getPosition() < 5)
   {
      this.setHeight(3); this.right(); this.downBy(3);
   }
}
```

[10 marks]

(ii) Subclassing is used to create a more specialised version of another class. Interfaces are usually used when classes are otherwise unrelated but their instances have a common set of messages to which they must respond. [2 marks]

Question 23 [15 marks]

(i)
```
public int compareTo(HoverFrog that)
{
   return that.getHeight() - this.getHeight();
}
```

[3 marks]

(ii)
```
private String[] rankNames;
private List<HoverFrog> rankableHoverFrogs;
private Map<String, HoverFrog> finalRankings;
```

[3 marks]

(iii)
```
public HoverFrogPond (List<HoverFrog> externalHoverFrogs,
                      String[] externalRankNames)
{
   super();
   Collections.sort(externalHoverFrogs);
   this.rankableHoverFrogs = externalHoverFrogs;
   this.rankNames = externalRankNames;
   this.finalRankings = new HashMap<String, HoverFrog>();
}
```

[3 marks]

(iv)
```
public void populateFinalRankings()
{
   int i = 0;
   while ((i < this.rankNames.length) && (i <this.rankableHoverFrogs.size()))
   {
      this.finalRankings.put(this.rankNames[i],
                             this.rankableHoverFrogs.get(i));
      i++;
   }
}
```

[6 marks]

Question 24 [20 marks]

(i) Serialisation is a much easier way of achieving persistence than explicitly writing the details of an object as characters to a text file. However, in order to recover a serialised object the Java Virtual Machine must know where to find the compiled class of the object (i.e. its `.class` file).

This can be contrasted with saving the details of an object (values of the instance variables) as characters in a text file – the file can be read by a human, and provided you tell them how the information is set out in the file (for example, account holder followed by account number followed by the balance) they will be able to make use of that information (if only by reading) without having the original `Account` class. [3 marks]

(ii)
```
public static void saveObject(Object anObject)
{
   String pathname = OUFileChooser.getFilename();
   File aFile = new File(pathname);
   ObjectOutputStream outStream = null;
   try
   {
      outStream = new ObjectOutputStream
            (new BufferedOutputStream(new FileOutputStream(aFile)));
      outStream.writeObject(anObject);
   }
   catch (Exception anException)
   {
      System.out.println("Error: " + anException);
   }
   finally
   {
      try
      {
         outStream.close();
      }
      catch (Exception anException)
      {
         System.out.println("Error: " + anException);
      }
   }
}
```
[13 marks]

(iii) (a)
```
inStream = new ObjectInputStream
            (new BufferedInputStream(new FileInputStream(aFile)));
```

(b)
```
anObject = inStream.readObject();
```
[4 marks]

[END OF SPECIMEN SOLUTIONS]