Candidate Instructions

Assignment - 205 Version 2.1

Create Software Components Using Java - Level 2

Assignment 7262-22-205 **Create Software Components Using Java Level 2**

Candidates are advised to read all instructions carefully before starting work and to check with your assessor, if necessary, to ensure that you have fully understood what is required.

You must, at all times, observe all relevant Health and Safety precautions.

Time allowed 6 hours

Introduction

This assignment is broken down into 3 parts:

1. A scenario is provided for candidates in the form of a company specification for a new product.
2. Task A provides a detailed design specification that should be followed by candidates when developing their program.
3. Task B provides presentation criteria that should be followed by candidates when producing their work.

Scenario

A games software development company, GameChoice, is developing a games program for use on the Internet. It will be run using a Java applet. The applet will allow a user to play the classic game of Hits and Misses in which the player must guess the colours of pegs.

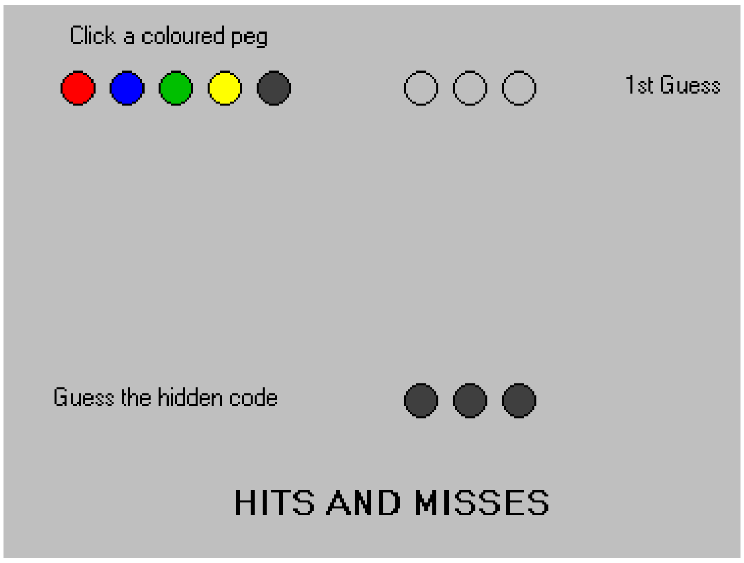
The outline specification for the applet states that

- the program randomly selects three colours from five possible colours
- the player has 6 guesses to work out which three colours in the correct sequence have been chosen by the program
- the program helps the player after each guess in terms of 'hits' (correct colours in the correct position) and 'misses' (colours which are correct, but not in the correct position)
- the game continues until either the player has solved the coloured code or has had 6 guesses
- if the player guesses the code before they have lost 6 lives, they have won
- if the player loses all their lives before they guess the coloured code, they have lost.

Task A

Candidates should use the following detailed specification to fulfil the company's requirements:

1. Create a HitMiss applet that has a GUI similar to that shown in the following picture.



2. The playing area is light gray in colour.

3. The title HITS AND MISSES must be displayed at the bottom with the text in blue and a large font size.

4. The 5 playing colours are red, blue, green, yellow and black and are displayed as filled circles below the instruction 'Click a coloured peg'.

5. The coloured code generated by the program is displayed as 3 filled black circles next to 'Guess the hidden code'.

6. The guess line is displayed as the guess number alongside 3 open circles.

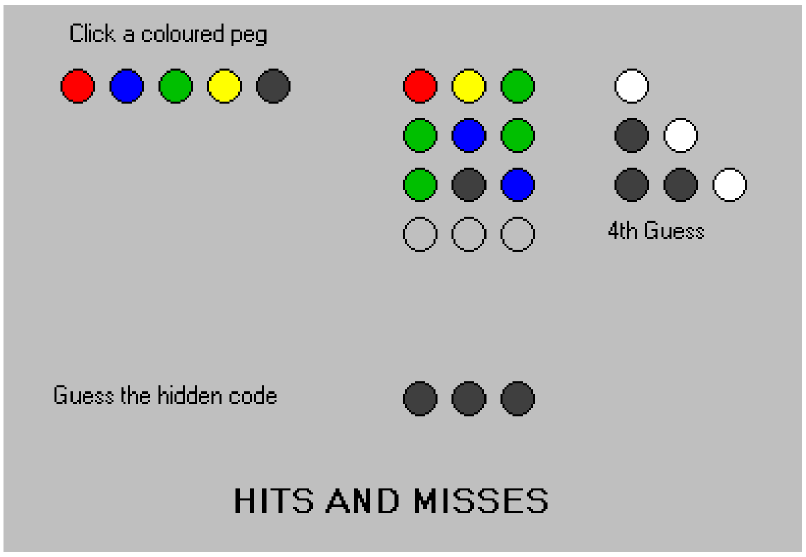
7. When the applet starts a random sequence of 3 colours from the 5 available is produced. Duplication of colours in the sequence is allowed.
   The following code for a class called RandCol has been produced by your manager to demonstrate the use of the Random class. Three colours are chosen randomly from a choice of five and three circles are drawn in an applet using the three randomly chosen colours.

```
import java.awt.*;
import java.util.*;

public class RandCol extends java.applet.Applet {
public void paint(Graphics g) {
int i, j, count=1;
Date dateObject = new Date();
long startseed;
int RandColour, PegColour1=0, PegColour2=0, PegColour3=0, PegColour4=0;
```

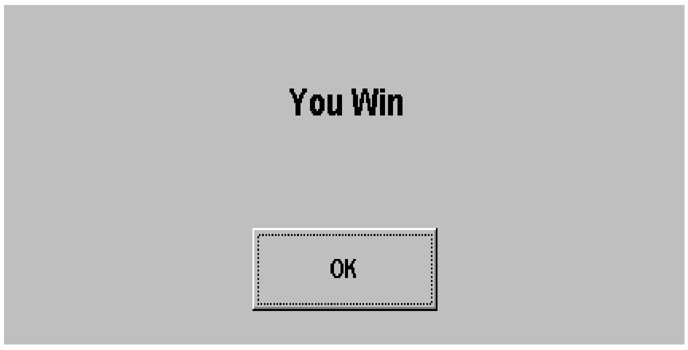
```
// The peg colours are held as numbers
// red is 0, blue is 1, green is 2, yellow is 3, black is 4
// set up a random object based on a starting seed of time
        startseed = dateObject.getTime();
        Random rand = new Random(startseed);

        while (count < 4){
        // loop three times until the three colours for the pegs have been randomly
        // chosen
        // get the next random number generated which is between 0-4
                RandColour= rand.nextInt(5);
                switch (count){
                case 1:
                        if (PegColour1 == 0){
                                PegColour1 = RandColour;
                                count++;
                        }
                        break;
                case 2:
                        if (PegColour2 == 0) {
                                PegColour2 = RandColour;
                                count++;
                        }
                        break;
                case 3:
                        if (PegColour3 == 0) {
                                PegColour3 = RandColour;
                                count++;
                        }
                        break;
        }
    }
        // set up the colour for first peg
      switch(PegColour1) {
                case 0:
                        g.setColor(Color.red);
                        break;
                case 1:
                        g.setColor(Color.blue);
                        break;
                case 2:
                        g.setColor(Color.green);
                        break;
                case 3:
                        g.setColor(Color.yellow);
                        break;
                case 4:
                        g.setColor(Color.black);
                        break;
}
        j=5; // j is the horizontal position in the applet
        i=60;// i is the vertical position in the applet
        // draw the first peg
        g.fillOval(j,i,20,20);
        g.setColor(Color.black);
        g.drawOval(j-1,i,20,20);
        // set up the colour for second peg
        switch(PegColour2) {
        case 0:
                g.setColor(Color.red);
                break;
        case 1:
                g.setColor(Color.blue);
                break;
        case 2:
                g.setColor(Color.green);
                break;
        case 3:
                g.setColor(Color.yellow);
                break;
        case 4:
                g.setColor(Color.black);
                break;
}
        // draw the second peg
        g.fillOval(j+30,i,20,20);
        g.setColor(Color.black);
        g.drawOval(j+29,i,20,20);
        // set up the colour for the third peg
        switch(PegColour3) {
        case 0:
                g.setColor(Color.red);
                break;
        case 1:
                g.setColor(Color.blue);
                break;
        case 2:
                g.setColor(Color.green);
                break;
        case 3:
                g.setColor(Color.yellow);
                break;
        case 4:
                g.setColor(Color.black);
                break;
        }
        // draw the third peg
        g.fillOval(j+60,i,20,20);
        g.setColor(Color.black);
        g.drawOval(j+59,i,20,20);
    }
}
```

8. The player selects a colour by clicking a coloured peg with the mouse. When a colour is selected the next open circle in the guess line is replaced with a filled circle of the selected colour. The open circles are replaced from left to right.

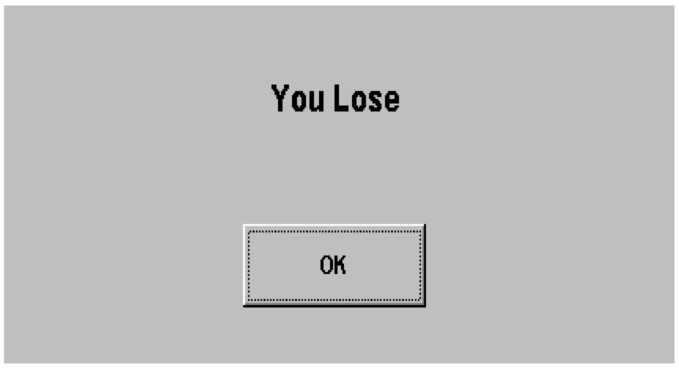
9. When 3 colours have been selected the program displays how many 'hits' (correct colour in the correct position) and 'misses' (correct colour but not in correct position) have been scored. Each 'hit' should be displayed as a filled black circle and each 'miss' as a filled white circle on the same line as the guess. The guess line (the next guess number together with three open circles) should be displayed on the next line (see below which shows a game after 3 guesses).



10. If the player guesses the code in 6 or less guesses then a dialog box should appear with 'You Win' in the centre in a large blue font. An OK button should be used to allow the player to return to the game.



11. If the player has not solved the code after 6 guesses then a dialog box should appear with 'You Lose' in the centre in a large red font. An OK button should be used to allow the player to return to the game.



12. The correct code should be revealed by replacing the black filled circles for the hidden code with the correct colours at the end of the game. The text 'Guess the hidden code' should be replaced with 'The hidden code is'.

13. Test the applet, check the expected results with the actual results and resolve any logical or run-time errors.

14. Print a program listing.

15. Produce a printout of the HitMiss applet (screen print)

16. Save the program on a disk *(the program should be saved under a meaningful name)*.

Task B

Candidates should follow the below criteria when producing their work:

1. The program conforms to the design specification.

2. The program uses the most appropriate data type(s).

3. Meaningful names are used when declaring variables.

4. The program syntax is consistently indented to aid readability.

5. The program is commented.

Note

- Candidates should produce the following for their assessor:
  - a printed program listing.
  - printout of the HitMiss applet (screen print)
  - a disk containing the program.
- At the conclusion of this assignment, hand all paperwork and disks to the test supervisor.
- Ensure that your name is on the disk (if using a floppy disk) and all documentation.
- If the assignment is taken over more than one period, all floppy disks and paperwork must be returned to the test supervisor at the end of each sitting.