INTPROG TB2: OBJECT ORIENTED PROGRAMMING

Practical Worksheet 7: Non-Collection Iteration

1. Introduction

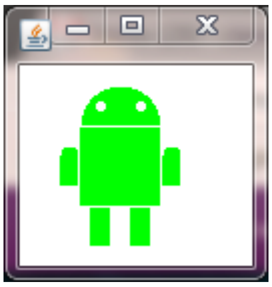
The practical session this week will be focusing on iteration which does not use collections. We will be predominantly using for loops, while loops, and do while loops. However, there may be cases in the example and the exercises where collection looping is required in addition to those which are the focus of the worksheet.

During the example, not all the code has been provided. There are some places where you will be expected to complete the method, or the class. You must complete this code, otherwise the example will not work as it should.

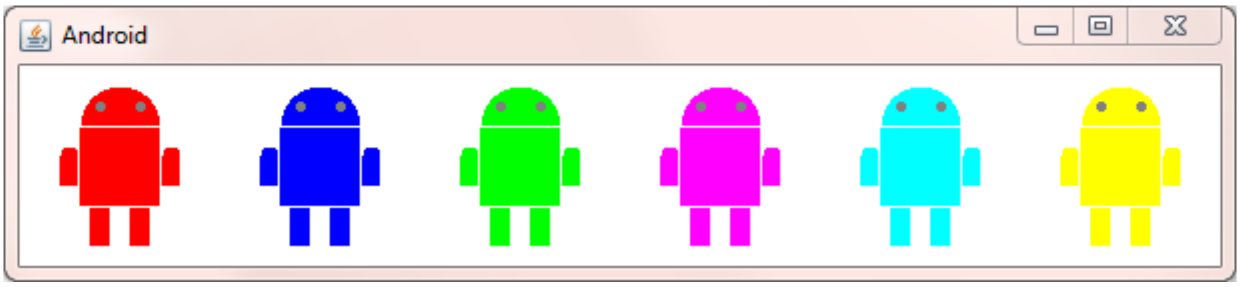
As usual, work through this practical session at your own speed. Make sure you understand everything included within the example before you move onto the questions at the end of the worksheet. If you have questions, either ask your practical tutor, or ask a question using the anonymous question form on Moodle.

2. Example – Android

In this practical session, we will extend the Android which was initially created in Lecture 9. On Moodle, you will find the Android Example starter files. Download the project package and open it either in BlueJ or NetBeans (make sure you choose the right download for your IDE). In its current state, running the main method will produce the following:



We are going to adapt this class so that it eventually produces the following:



The colour of all the Androids will change every 50 milliseconds, using a while loop. The while loop will run for 100 iterations, then stop.

Firstly we want to adapt the android class so that it stores the following additional information:

- Current Colour of the Android {variable name: currentColour, type: Color}
- X Coordinate of the starting position of the android (this will be the top left point of the enclosing 100x100 square) {variable name: topLeftX, type: double}
- Y Coordinate of the starting position of the android (this will be the top left point of the enclosing 100x100 square) {variable name: topLeftY, type: double}

Create the instance variables for these, and update the constructor so that they are assigned values based on the parameters. Also, write the accessor and mutator method for the currentColour variable. This will complete the Android class.

Create a new class called "AndroidPatchwork". Declare the following instance variables:

- An ArrayList of Androids (name: androids, type: ArrayList<Android>)
- The window for the androids to be drawn on (name: canvas, type: Canvas)
- An ArrayList of colours (name: colourList, type: ArrayList<Color>)
- The number of androids drawn in the window (name: numAndroids, type: int)

Write the constructor for the AndroidPatchwork. The colourList will have 6 color objects: Color.RED, Color.BLUE, Color.GREEN, Color.MAGENTA, Color.CYAN, and Color.YELLOW.

```java
/**
 * Constructor for the AndroidPatchwork
 * @param win the window to draw the androids in
 * @param numberAndroids  the number of androids to draw
 */
public AndroidPatchwork(Canvas win, int numberAndroids)
{
    androids = new ArrayList<Android>();
    canvas = win;
    colourList = new ArrayList<Color>();
    colourList.add(Color.RED);
    colourList.add(Color.BLUE);
    colourList.add(Color.GREEN);
    colourList.add(Color.MAGENTA);
    colourList.add(Color.CYAN);
    colourList.add(Color.YELLOW);
    numAndroids = numberAndroids;
}
```

Next we need to write a method which will initially draw all the androids in the window. The method uses a for loop to create the number of androids which are required, based on the number entered in the constructor (and stored as an instance variable). The androids are all stored in an ArrayList, so that we can use them later. Once all the required androids are created, it will use a for-each loop to draw them all in the window.

```java
/**
 * Method to draw all the Androids in the window
 */
public void drawAndroids()
{
    for (int i = 0; i < numAndroids; i++)
    {
        androids.add(new Android(canvas, colourList.get(i), i*100, 0));
    }

    for(Android android : androids)
    {
        android.drawAndroid();
    }
}
```

The purpose of this example is to create Androids which change colours – I've called this a disco cycle. To make it loop 100 times, we will be using a while loop which counts the iterations. The method then uses a for-each loop to work through each of the Androids on the screen to change their colour. To decide which colour, we need an additional method which will calculate the next colour in the sequence.

```
/**
 * A method to get the next colour in the sequence for the android
 * @param currentColour the current colour of the android
 * @return the next colour in the sequence for the android
 */
public Color nextColour(Color currentColour)
{
    int currentIndex = colourList.indexOf(currentColour);
    int newIndex = (currentIndex + 1) % numAndroids;

    return colourList.get(newIndex);
}


/**
 * Method to make the androids change colour in a predefined cycle
 */
public void discoAndroid()
{
    Color currentColour;
    Color newColour;
    int iterationNo = 0;

    while(iterationNo < 100)
    {
        for(Android android : androids)
        {
            currentColour = android.getCurrentColour();
            newColour = nextColour(currentColour);
            android.setCurrentColour(newColour);
            android.drawAndroid();
            canvas.wait(50);   // wait 50 milliseconds before continuing the for-each loop
        }

        iterationNo++;
    }

}
```

This is the AndroidPatchwork class completed. All that is left now is to write the GetInputs class, which will get the number of androids which need to be drawn from the user. In addition, it will be used to define the start of the discoAndroid method.

Start by creating a GetInputs class, with a single instance variable: a KeyboardInput object called "keyboardInput". The constructor for this class simply initialises the keyboardInput variable.

```
/**
 * Constructor for GetInputs class
 */
public GetInputs()
{
    keyboardInput = new KeyboardInput();
}
```

The first method we are going to need is one which gets the user to input the number of androids they want to draw. This needs to be a number between 1 and 6. Therefore, we are going to use a do-while loop to check the validity of the entry by the user, and keep asking for a number until it is a valid input.

```java
/**
 * Method to get the number of androids the user wants to draw.  It must be between 1 and
 * 6 androids
 * @return the number of androids to draw on the screen
 */
public int getNumAndroidsFromUser()
{
    int numAndroids;
    boolean validEntry = false;

    do
    {
        System.out.print("Enter the number of Androids to Draw: ");
        numAndroids = keyboardInput.getInputInteger();

        if(numAndroids <= 0 || numAndroids > 6)
        {
            System.out.println("Sorry, the number must be between 1 and 6 (inclusive)");
            System.out.println("Please try again");
        }
        else
        {
            validEntry = true;
        }

    }while(!validEntry);

    return numAndroids;
}
```

The final method which needs to be written in the GetInputs class is the getDiscoStart() method. This will be used to start the androids changing colour. The method doesn't need to return anything, it essentially waits for the enter key to be pressed. Once it has, the discoAndroid method from the AndroidPatchwork class can be run (this will be managed in the main method).

```java
/**
 * Method to detect when to start running the android disco method
 */
public void getDiscoStart()
{
    System.out.println("Press enter to start the disco");
    keyboardInput.getInputString();
}
```

Finally, we need to write the main method which will get the number of androids, draw them, wait for the enter key, and then run the discoAndroid method.

```
public static void main(String[] args)
{
    GetInputs getInputs = new GetInputs();

    int numAndroids = getInputs.getNumAndroidsFromUser();
    int widthAndroids = numAndroids * 100;

    Canvas canvas = new Canvas("Android",widthAndroids, 100, Color.WHITE);

    AndroidPatchwork patchwork = new AndroidPatchwork(canvas, numAndroids);
    patchwork.drawAndroids();

    getInputs.getDiscoStart();
    System.out.println("Disco started");
    patchwork.discoAndroid();

}
```

When you run the main method, try entering invalid inputs (to check the getNumAndroidsFromUser method is working). Then press the enter button to get the disco to run.

3. Practice Questions

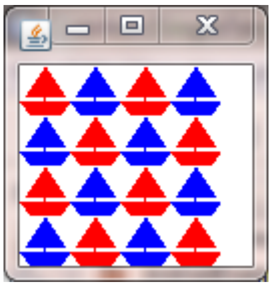
The following questions are designed to give you some practice at using loops without a collection in Java. If you have any questions, please make sure you ask the members of staff within the practical sessions.

1. Using a nested for loop, write method which will take an integer as a parameter, and draws a number square. The following square will be drawn when the method is run with a parameter value of 4:

```
4 3 2 1
5 4 3 2
6 5 4 3
7 6 5 4
```

2. Extend the number square program so that the user is asked to enter a number between 1 and 5 (inclusive). The appropriate number square should then be drawn using the input provided by the user.

3. Write a class which will draw the following patch:



4. Extend the program so that the user can enter two valid colours (separately) to draw the patch in. The valid colours are as follows:
   (a) For colour1: red, blue, and green
   (b) For colour2: magenta, yellow, and cyan.

   Hint: use the getInputString method for the user to enter a letter representing a colour, then use following to make 1st letter entered a capital character:

```
char colourCharacter = Character.toUpperCase(enteredColour.charAt(0));
```

   The colourCharacter can then be used within the if statements to set the appropriate colour.

5. Extend the program written in the previous question to include a class called "Patchwork" which will take the following 5 parameters in the constructor:

   - The window to draw the patches in
   - The first colour of the patch
   - The second colour of the patch
   - Number of rows in the patchwork
   - Number of columns in the patchwork

   Within this class, write a method which fills a row x column sized window with patches using colours which has been entered by the user (before the patchwork object is initialised).
   Hint: You will need to adapt your patch class so that it takes an x and y starting point.

6. Extend your program so that the user can enter the number of row and columns for the window. You will need to limit the number of rows and columns to a maximum of 6 for each.

7. Extend your program so that the user can enter a row and column index for a patch. This value will then be used to switch the colour1 and colour2 variables, to change the order of colours for the single patch chosen. The program will also need to ask the user whether they want to continue switching the colours for their patches.