Tutorial 6: Iteration

This tutorial will provide practice at analysing and implementing iterations.

Task 1: Sum of positive and negative 

Write a program to read in 10 integer values and output 2 numbers:
the sum of all the positive integers
the sum of all the negative integers

Step 1: Analyse the problem

Using the techniques covered in Lecture 7 (slide 5), analyse the above problem and store your analysis in a file called `SumPosNegAnalysis.doc`

The analysis should consider:

- how many times the loop is repeated?
- what operations are done before the loop?
- what operations are done inside the loop?
- what operations are done after the loop?

Step 2: Create a NetBeans project

- create a new project called `SumPosNegProj` in a folder called `T6`

Step 3: Write source code

- add a new file called `SumPosNeg` to the `SumPosNegProj` project
- using the analysis from step 1, encode the solution

Step 4: Test your program and take screen shots

- run your program
- take a screen shot of the output and save it as a file in your project folder called `SumPosNeg.jpg`

Portfolio requirements

- The NetBeans project for this completed task
- `SumPosNegAnalysis.doc` from step 1
- `SumPosNeg.jpg` from step 4, containing screen shots

Task 2: Grid 

Write a program to prompt the user to enter 2 numbers: `width` and `height` and uses a pair of nested loops to output a grid of asterisks with dimensions `width * height`.

There is no requirement to validate the input data to ensure it is positive.

For example, given `width = 3` and `height = 4`, the program should output the following grid:

```
* * *
* * *
* * *
* * *
```

Step 1: Create a NetBeans project

- create a new project called `GridProj` and store it in a folder called `T6`

Step 2: Write source code

- add a new file called `Grid` to the `GridProj` project
- implement the program to output the grid of asterisks

Step 3: Test your program and take screen shots

- run your program with `width = 3`, `height = 4`
- take a screen shot of the output and save it in your project folder as `Grid.jpg`

Portfolio requirements

- The NetBeans project for this completed task
- `Gride.jpg` from step 3, containing a screen shot of the output

Task 3: Average of positive and negative

Extend task 1 that calculates the sum of the positive numbers and the sum of the negative numbers input so that it also calculates and outputs the average of the positive numbers and the average of the negative numbers.

Appropriate messages should be output if there are no positive numbers or if there are no negative numbers.

Step 1: Create a NetBeans project

- create a new project called `AveragePosNegProj` in a folder called `T6`

Step 2: Extend source code

- add a new file called `AveragePosNeg` to the `AveragePosNegProj` project
- copy the code from `sumPosNeg` and copy it to your new file
- add code so that the program calculates and outputs the average of the positive numbers and the average of the negative numbers

Step 3: Test your program and take screen shots

- run your program with all positive numbers and ensure that it works correctly
- run your program with all negative numbers and ensure that it works correctly
- run your program with a mix of numbers and ensure that it works correctly
- take a screen shot of each test run and save it as a separate file in your project folder called `AveragePosNeg1.jpg`, `AveragePosNeg2.jpg`, etc

Portfolio requirements

- The NetBeans project for this completed task
- `AveragePosNeg1.jpg`, `AveragePosNeg2.jpg`, etc from step 3, containing screen shots

Task 4: Triangle 

Modify task 2 that outputs a grid of asterisks of a required size so that it outputs a triangle of asterisks of a given number of rows.

For example, given `rows = 4`, the program should output the following triangle:

```
*
* *
* * *
* * * *
```

Step 1: Create a NetBeans project

- create a new project called `TriangleProj` and store it in a folder called `T6`

Step 2: Extend source code

- add a new file called `Triangle` to the `TriangleProj` project
- using the code from `Grid.java` as a guide, write code to output a triangle given the number of rows required

Step 3: Test your program and take screen shots

- run your program with `rows = 4`
- take a screen shot of the output and save it in your project folder as `Triangle.jpg`

Portfolio requirements

- The NetBeans project for this completed task
- `Triangle.jpg`, from step 3, containing a screen shot of the output

Task 5: Isosceles Triangle 

Write an application that uses loops to output an inverted isosceles triangle where the number of rows is defined by the user.

For example, given `rows = 4`, the program should output the following triangle:

```
* * * *
 * * *
  * *
   *
```

Portfolio requirements

- The NetBeans project for this completed task
- `Isosceles.jpg,` containing a screen shot of the output

These tasks will provide practice at analysing and implementing non-deterministic iterations.

Task 6: Type of loop 

Step 1: Step 1: Decide loop types

State whether each of the following is deterministic or non-deterministic and justify your choice:

a) writing Christmas cards to family
b) washing up
c) reading a book
d) waiting for someone to answer the phone
e) counting to 10 in French
f) playing a game of 501 in darts
g) laying the table for a dinner party
h) answering this question

NB: some of these could be either deterministic or non-deterministic

Store your answers in a Word document called `LoopTypes.doc` in the `T6` folder within the `IJ` folder.

Portfolio requirements – required:

- `LoopTypes.doc` containing the list of scenarios, whether they are deterministic or non-deterministic and your justification

Task 7: Dice match 

Implement a program to simulate throwing 2 dice and output how many throws there were before the dice matched. The values on the pairs of die should be output for each throw, including the matching pair.

Step 1: Analyse the problem

Using the techniques covered in the lecture, analyse the above problem and store your analysis in a file called `DiceMatchAnalysis.doc`

The analysis should consider:

- how many times the loop is repeated?
- what operations are done before the loop?
- what operations are done inside the loop?
- what operations are done after the loop?

Step 2: Create a NetBeans project

- create a new project called `DiceMatchProj` in a folder called `T6`

Step 3: Write source code

- add a new file called `DiceMatch` to the `DiceMatchProj` project
- using the analysis from step 1, encode the solution

Step 4: Test your program and take screen shots

- run your program
- take a screen shot of the output and save it in your project folder as `DiceMatch.jpg`

Portfolio requirements

- The NetBeans project for this completed task
- `DiceMatchAnalysis.doc` from step 1, containing your analysis of the problem
- `DiceMatch.jpg` from step 4, containing a screen shot of the output

Task 8: Validate percentage 

Implement a program that prompts the user to input a percentage in the range 0 – 100. If they input a value outside this range, an error message and a prompt to re-enter the data should be output. This should continue until they input a valid value which is then displayed.

Step 1: Create a NetBeans project

- create a new project called `ValidatePercentProj` in a folder called `T6`

Step 2: Write source code

- add a new file called `ValidatePercent` to the `ValidatePercentProj` project
- write code to prompt the user to input a percentage until a valid value is entered

Step 3: Design a test plan

- design a test plan to test your program and store it in `ValidatePercentTest.doc`

Step 4: Test your program

- run your program with each test case you designed in your test plan
- take a screen shot of the output and save it in your project folder called `ValidatePercent.jpg`

Portfolio requirements

- The NetBeans project for this completed task
- `ValidatePercentTest.doc` from step 3, containing your test plan
- `ValidatePercent.jpg` from step 4, containing a screen shot of the output

Task 9: Shapes 

Write an application that prompts the user to choose what kind of shape is output, and to define its dimension. The shape options are:

- Triangle
  - Left-aligned, right-angled
  - Right-aligned, right-angled
  - Isosceles
  - Inverted isosceles
- Rectangle

The selected shape is to have only its outline drawn. For example, if the user chooses an inverted isosceles triangle with five rows, the program should output:

```
* * * * *
 *     *
  *   *
   * *
    *
```

The program continues asking the user what shape to draw until the user picks the exit option.

Portfolio requirements

- The NetBeans project for this completed task
- Image files showing all possible outputs from your application