Tutorial 5: Selection

This tutorial will provide practice at analysing, implementing and testing selections.

Task 1: Conditions 

Assuming a = 5, b = 4, c = 3, indicate whether each of the following conditions are true or false:

a) (a < b)
b) (a != b)
c) (a > b) && (a < c)
d) (a > b) || (a < c)
e) (a – 1 == b)
f) !(b > c)

Store your answers in a Word document called `Conditions.doc` in the `T5` folder within the `IJ` folder.

Portfolio requirements

- `Conditions.doc` containing list of conditions and whether you think they are true or false

Task 2: Odd or even 

Write a program to input an integer and output whether it is odd or even. Ensure you run your program with both an odd number and an even number.

Hint: use the remainder operator to check whether the number is divisible by 2

Step 1: Analyse the problem

Using the techniques covered in Lecture 5, analyse the above problem and store your analysis in a file called `OddOrEvenAnalysis.doc`

The analysis should consider the following:

- what data is used?
- is all data dealt with in same way?
- what operations are done before the selection?
- what operations are done for each possibility?
- what operations are done after the selection?

Step 2: Create a NetBeans project

- create a new project called `OddOrEvenProj` in a folder called `T5`

Step 3: Write source code

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

Step 4: Test your program

- run the program with an odd number
- run the program again with an even number

Step 5: Take a screen shot of the output

- take a screenshot of the output from each run and store them in your project folder as `OddOrEven1.jpg` and `OddOrEven1.jpg`

Portfolio requirements

- The NetBeans project for this completed task
- `OddOrEvenAnalysis.doc` from step 1, containing the analysis of the problem
- `OddOrEven1.jpg` and `OddOrEven2.jpg` from step 5, containing screen shots of each test run

Task 3: Password 

Write a program to prompt the user to enter their password as a string. If the password matches the password hard-coded in the program regardless of case, output a welcome message.

Step 1: Analyse the problem

Using the techniques covered in Lecture 5, analyse the above problem and store your analysis in a file called `PasswordAnalysis.doc`

The analysis should consider the following:

- what data is used?
- is all data dealt with in same way?
- what operations are done before the selection?
- what operations are done for each possibility?
- what operations are done after the selection?

Step 2: Create a NetBeans project

- create a new project called `PasswordProj` and store it in a folder called `T5`

Step 3: Write source code

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

Step 4: Test your program

- run the program with a valid password
- run the program again with an invalid password

Step 5: Take a screen shot of the output

- take a screenshot of the output from each run and store them in your project folder as `Password1.jpg` and `Password2.jpg`

Portfolio requirements

- The NetBeans project for this completed task
- `PasswordAnalysis.doc` from step 1 containing the analysis of the problem
- `Password1.jpg` and `Password2.jpg` from step 5, containing screen shots of each test run

Task 4: Theatre tickets 

A theatre sells tickets according to the following details:

Ticket	Stalls
Adult	£10.50
Child	£7.30
Concessions	£8.40

There are two offers that should be automatically applied:

- 1 free adult chaperone place for every group of 10 children, for example:
  - for up to 10 children, all adults must pay
  - between 10 and 19 children: up to 1 free adult
  - between 20 and 29 children: up to 2 free adults
- 10% discount if the bill total exceeds £100.00, after any free chaperone seats have been dealt with

If the tickets are to be collected in person, there is no additional charge, otherwise postage & packaging costs £2.34, and is added after all offers have been applied.

Write a program to prompt the user for ticket requirements and output the bill as a formatted receipt with details and final total.

Develop a full test plan to test your program.

Portfolio requirements

- The NetBeans project for this completed task
- `TheatreTickets.doc` containing your test plan
- Image files containing screen shots of the actual results when following your test plan

Task 5: Positive, negative or zero

Write a program to input an integer and output whether it is positive, negative or zero. Produce a test plan to fully test your program.

Step 1: Create a NetBeans project

- create a new project called `PosNegProj` in a folder called `T5`

Step 2: Write source code

- add a new file called `PosNeg` to the `PosNegProj` project
- implement a program to determine whether the input is positive, negative or zero

Step 3: Design a test plan

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

Step 4: Test your program and take screen shots

- run your program with each test case you designed in your test plan
- take a screen shot of each test run and save it as a separate file in your project folder called `PosNeg1.jpg`, `PosNeg2.jpg`, etc

Portfolio requirements

- The NetBeans project for this completed task
- `PosNegTest.doc` from step 3, containing your test plan
- `PosNeg1.jpg`, `PosNeg2.jpg`, etc from step 4, containing screen shots

Task 6: Grade 

Extend the program from lecture 6 (slide 15: outputs a grade based on a percentage) so that the integer input by the user is validated to be a percentage in the range 0-100 before the grade is determined and output.

```
//output grade based on percent
if (percent >= 70)
{
       System.out.println("A");
}
else if (percent >= 60)
{
       System.out.println("B");
}
else if (percent >= 50)
{
       System.out.println("C");
}
else if (percent >= 40)
{
       System.out.println("D");
}
else
{
       System.out.println("E");
}
```

Step 1: Create a NetBeans project

- create a new project called `GradeProj` in a folder called `T5`

Step 2: Extend source code

- add a new file called `Grade` to the `GradeProj` project
- copy the above code from Blackboard and paste it into the `Grade` file
- extend the code so that it validates the input before determining the grade

Step 3: Test your program and take screen shots

- run your program with the following values: -1, 0, 100, 101
- take a screen shot of each test run and save it as a separate file in your project folder as `Grade1.jpg`, `Grade2.jpg`, etc

Portfolio requirements

- The NetBeans project for this completed task
- `Grade1.jpg`, `Grade2.jpg`, etc from step 3, containing screen shots of each test run

Task 7: Post 

Write a program to input the type of letter received as a string (either "bill", "circular", "postcard" or "letter") and output what to do with it as follows:

- bills must be paid
- circulars are thrown away
- postcards are put on the wall
- personal letters are read and have replies written for them

The program should also output an error message if the letter type is not recognized.

Step 1: Create a NetBeans project

- create a new project called `PostProj` in a folder called `T5`

Step 2: Write source code

- add a new file called `Post` to the `PostProj` project
- encode the program to output what to do with various types of post

Step 3: Design a test plan

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

Step 4: Test your program and take screen shots

- run your program with each test case you designed in your test plan
- take a screen shot of each test run and save it as a separate file in your project folder called `Post1.jpg`, `Post2.jpg`, etc

Portfolio requirements

- The NetBeans project for this completed task
- `PostTest.doc` from step 3 containing your test plan
- `Post1.jpg, Post2.jpg,` etc from step 4, containing screen shots