Tutorial 2: Variables & data types

This tutorial will develop your programming skills by starting to work with numeric variables and calculations.

Task 1: Break even 

The break-even point for a business is defined as the point where the profit equals the expenses. The following values are required to calculate this:

- cost to produce each item
- price per item
- overhead expenses

Dividing the overhead expenses by the profit per item (the difference between the cost and the price) gives the number of items that must be sold before "breaking even".

For example, if it costs £2.00 to produce an item which is then priced at £3.50, there is £1.50 profit per item. If the overhead expenses are £6.00, the company would need to sell 4 items to break even.

Hint: The profit per item and the number of items that must be sold to break even should be calculated and stored. The number of items should then be output.

Step 1: Analyse the problem

- decide what type of data is to be used
- using the notes given in Lecture 2 (slide13), analyse the problem and save it in a word document called `BreakEvenAnalysis.doc`

Step 2: Create a NetBeans project

- using the information given in Lecture 1, create a new folder called `T2` within your `IJ` folder on the H drive
- using NetBeans create a new project called `BreakEvenProj` in your `T2` folder

Step 3: Write source code

- add a new file called `BreakEven` to the `BreakEvenProj` project
- using your analysis from step 1, write the `BreakEven` code, to output the break-even point for the values given above
- ensure that you use meaningful names for all variables

Step 4: Run and test your application

- run your `BreakEven` application
- does it give the output you were expecting? If not, locate and correct the logical errors in your source code
- modify the code to use different values for the item cost, item price, and overhead expenses. Does your program still give the correct answers?
- modify the code to use the original values from the specification and run it again

Step 5: Take a screen shot of the output

- press [ALT + Print Scrn]
- open Paint (Start | Programs | Accessories | Paint)
- paste the screenshot using [CTRL + V]
- save the image in your project folder as `BreakEven.jpg`

Portfolio requirements:

- `BreakEvenAnalysis.doc` from step 1, containing your analysis of the problem
- The NetBeans project for this task (the `H:/IJ/ T2 /BreakEvenProj` folder)
- `BreakEven.jpg` from step 5, showing output from `BreakEven.java`

Task 2: Data types 

Step 1: Decide data types

State the most appropriate data type for the following data and justify your choice:

a) a company's profit
b) a person's weight
c) a book title
d) average rainfall in June
e) number of goals scored by a footballer
f) a telephone number, including area code
g) a shirt size

Store your answers in a Word document called `DataTypes.doc` in the `T2` folder within the `IJ` folder.

Portfolio requirements

- `DataTypes.doc` containing the list of appropriate data types and justification for their selection

Task 3: Savings 

With integer division it is important to test that the correct calculation is performed.

Step 1: Copy source code

- using NetBeans create a new project called `SavingsProj` in the `T2` folder within your `IJ` folder
- add a new file called `Savings` to the `SavingsProj` project
- download the following code from Blackboard, and copy & paste it in the `Savings.java` file
- this code should calculate 25% of my savings and display it in the output panel

```
//calculate and output half of my savings
public class Savings
{
    //your share is 25% of my savings
    public static void main (String args[])
    {
        int mySavings = 3000;
        int yourPercentage = 25;

        int yourShare = (mySavings * (yourPercentage / 100));

        System.out.println("Your share: " + yourShare);
    }
}
```

Step 2: Run the program and take a screen shot of the output

- run the program
- are the results as expected?
- save an image of the output in your project folder as `SavingsIncorrect.jpg`

Step 3: Correct the code and take a screen shot of the output

- modify the code so that your share is correctly calculated as 750; you must not change the data type of any of the variables
- run the program
- save an image of the output in your project folder as `SavingsCorrect.jpg`

Portfolio requirements

- The NetBeans project for this completed task (the `H:/IJ/T2/SavingsProj` folder)
- `SavingsIncorrect.jpg` from step 2, showing incorrect output from `Savings` program
- `SavingsCorrect.jpg` from step 3, showing correct output from `Savings` program

Task 4: Sweets 

A teacher has bought a packet of 25 sweets that she is going to share out equally between her 7 students. Because a single sweet cannot be shared, the teacher will keep what is not given to the children. The teacher will keep the minimum number of sweets possible.

Write a program to determine and output the number of sweets each child will receive. As a single sweet cannot be shared between pupils, the program must calculate the number of sweets per child and the number that the teacher keeps for herself.

Step 1: Analyse the problem

- using the notes given in Lecture 2 (slide13), analyse the problem and save it in a word document called `SweetsAnalysis.doc`

Step 2: Create a NetBeans project

- using NetBeans create a new project called `SweetsProj` in your `T2` folder

Step 3: Write source code

- add a new file called `Sweets` to the `SweetsProj` project
- using your analysis from step 1, write the `Sweets` code, to calculate and output the number of sweets for the teacher and for each child

Hint: You will need to use integer division and the remainder operator

Step 4: Run and test your application

- run your `Sweets` application
- does it give the output you were expecting? If not, locate and correct the logical errors in your source code
- modify the code to use different values for the number of sweets and the number of children. Does your program still give the correct answers?
- modify the code to use the original values from the specification and run it again

Step 5: Take a screen shot of the output

- save a screenshot of the output in your project folder within the `IJ` folder as `Sweets.jpg`

Portfolio requirements

- `SweetsAnalysis.doc` from step 1, containing your analysis of the problem
- The NetBeans project for this task (the `H:/IJ/T2/SweetsProj` folder)
- `Sweets.jpg` from step 5, showing your program's output

Task 5: Cans of paint 

To complete this task, you will need to find out about the `Math` class. You could consult the on-line documentation at:

http://docs.oracle.com/javase/8/docs/api/java/lang/Math.html

You will need some of the methods from that class. For example, to call the `abs()` method, you type: `Math.abs(`*`a_number_or_variable`*`)`
e.g.: `Math.abs(num)`

Write a single program that uses methods from the `Math` class to address the scenario described below.

Scenario

A can of paint covers 28 m^2 of wall. Each can of paint has a diameter of 15 cm and a height of 30 cm. The shop that sells the paint helps customers by packing the cans into boxes with internal measurements of 0.60 x 0.30 x 0.35 metres (L x W x H).

Output:

1. The minimum number of cans that must be bought to paint the walls of a hall whose floor measures 25 x 30 metres, and whose ceiling is 3.6 metres above the floor (use an appropriate method from the `Math` class);
2. The number of full boxes given to the customer who buys this quantity of paint (use an appropriate method from the `Math` class rather than integer division);
3. The number of cans not packed into boxes.

Portfolio requirements:

- `CansOfPaintAnalysis.doc`, containing your analysis of the problem
- The NetBeans project for this task
- `CansOfPaint.jpg` showing your program's output using the values specified in the scenario