Duke

In all tutorials, the use of the Duke image is permitted as described at:
https://kenai.com/projects/duke/pages/Home

The **difficulty of tasks** is indicated by the number of Duke images next to the task name:

Straightforward – *everybody is expected to do these*

Intermediate – *not too hard, but might need help from your tutor*

Challenging – *may require you to do background research to complete the task*

The **expectation for completion** of tasks is as follows:

Everyone is required to complete all one-Duke tasks

Try to complete as many two-Duke tasks as possible

Attempt three-Duke tasks only if you have spare learning time left (remember, every week, you should work on IJ for 4 hours in class, and 6 hours out of class)

The **portfolio tests** will include a mix of tasks from all levels of difficulty in a ratio of approximately:

50% of questions

25% of questions

25% of questions

You will not know before the tests which tasks will be included.

NB: The tutorial work begins on the next page

Tutorial 1: Introduction

This tutorial assumes that you are using NetBeans.

Task 1: Hello World 

Step 1: Create a new folder called `IJ` and a subfolder called `T1`

- using either My Computer or Windows Explorer navigate to the H drive and create a new folder called `IJ`
- create a new subfolder within `IJ` called `T1`

Folder names can be as long as you like but the longer they are the more likely you are to make typos. When naming directories it is best to choose short but meaningful names.

NB: The following examples assume that you have created a folder called `IJ` containing a subfolder called `T1` (for Tutorial 1) on the H drive

Step 2: Create a NetBeans project

- using the information given in the lecture, create a new project called `HelloWorldProj`
- store it in your `T1` folder in the `IJ` folder on the H drive

Step 3: Write source code

- using the information given in the lecture, add a new file called `HelloWorld` to the `HelloWorldProj` project
- type the following code in the editor pane:

```
// output Hello World
public class HelloWorld
{
        public static void main (String[] args)
        {
                System.out.println ("Hello World");
                System.out.println ("Welcome to Java");
        }
}
```

- make sure that you use the correct brackets
- ensure you put semicolons in the right places
- get into the habit of putting a comment to explain what the application does at the start of the program

Step 4: Run program

- using the information given in the lecture (slide 26), run the `HelloWorld` application after correcting any errors that may occur

Step 5: Take a screen shot of the output

- ensure that the NetBeans screen is the active one by clicking on it
- press [ALT + Print Scrn]
- open Paint (Start | Programs | Accessories | Paint)
- paste the screenshot using [CTRL + V]
- save the image in your project folder as `HelloWorld.jpg`

Portfolio requirements

- The NetBeans project for this task (the entire `H:/IJ/T1/HelloWorldProj` folder)
- `HelloWorld.jpg` from step 5, showing output from `HelloWorld.java`

Task 2: Debug 

When implementing programs it is very easy to make typing mistakes. With practice, these mistakes become easier to spot.

NetBeans highlights syntax errors by underlying code in red and placing an exclamation mark in the column to the left of the incorrect line. More information about the error is displayed in a pop-up box when the cursor is either placed on the code or the exclamation mark.

To get a feel for the error messages you might encounter, this exercise requires you to remove errors from a piece of code.

Step 1: Create a NetBeans project

- create a new project called `DebugProj` in the `T1` folder within your `IJ` folder on the H drive

Step 2: Enter source code

- add a new Java class file called `Debug` to the `DebugProj` project
- download the following code from Blackboard, and copy & paste it into the `Debug.java` file
- this code should output two strings to the output panel

```
/error ridden program
public Class ErrorProne

    public Void Static Main (string () args}
    (
        system.Out.PrinT1n('Here is the first string to output'));
        System.out.println"Here is the second string to output";
}
```

Step 3: Take a screen shot of the code showing errors

- ensure that the NetBeans screen is the active one by clicking on it
- press [ALT + Print Scrn]
- open Paint (Start | Programs | Accessories | Paint)
- paste the screenshot using [CTRL + V]
- save the image in your project folder as `DebugWithErrors.jpg`

Step 4: Debug the code

- using the hints displayed by NetBeans, remove all errors from the code and run it

Step 5: Take a screen shot of the code with no errors

- save the image in your `T1` folder within the `IJ` folder as `DebugNoErrors.jpg`

Portfolio requirements

- The NetBeans project for this completed task (the `H:/IJ/T1/DebugProj` folder)
- `DebugWithErrors.jpg` from step 3, showing `Debug` program with errors
- `DebugNoErrors.jpg` from step 5, showing `Debug` program without errors