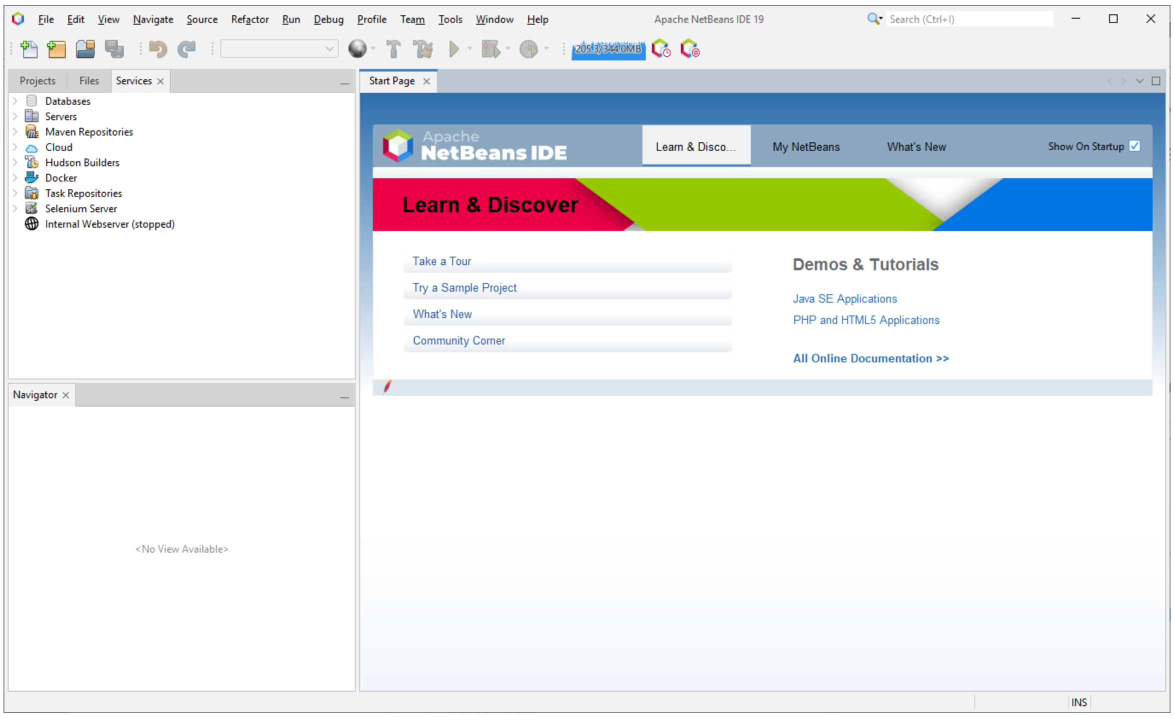
Dr Tomasz Kurowski
t.j.kurowski@cranfield.ac.uk

PRACTICAL 1 – FIRST JAVA PROGRAMS

This document will guide you through a simple demonstration of how you can create, build, and run a Java program using NetBeans.

Java programming using NetBeans

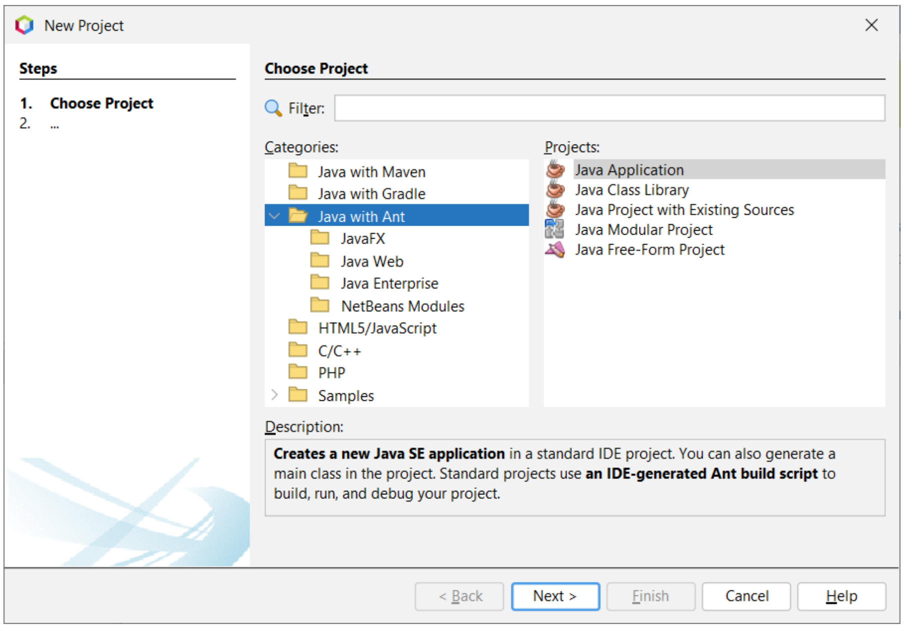
Launch the NetBeans IDE. If you are starting NetBeans for the first time, you might get a message about downloading JUnit and/or register to NetBeans - ignore them both! Once the IDE has started, you should see a screen something like the one below. Feel free to browse around, and click “Take a Tour”, although this may introduce a number of concepts you are not yet familiar with.



Start by creating a ‘New Project’ from the ‘File’ menu in order to create a new Java application in NetBeans.

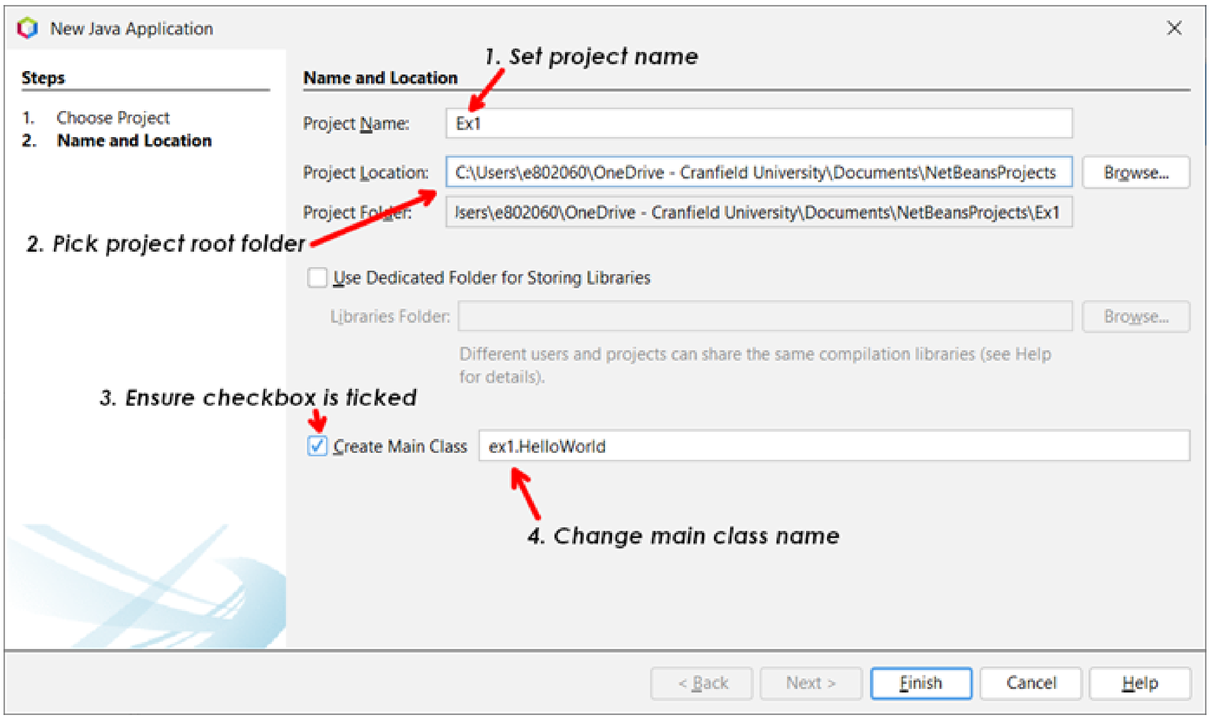
After clicking on New Project, the window shown below should appear. Choose **Java with Ant** and **Java Application** from the right hand- side panel. We are going to change a number of default options in order to properly configure the project.

Dr Tomasz Kurowski
t.j.kurowski@cranfield.ac.uk



Creates a new Java SE application in a standard IDE project. You can also generate a main class in the project. Standard projects use **an IDE-generated Ant build script** to build, run, and debug your project.

The next window dialog to pop-up requests additional information from you:



Project Name: Ex1
Project Location: C:\Users\e802060\OneDrive - Cranfield University\Documents\NetBeansProjects
Project Folder: lsers\e802060\OneDrive - Cranfield University\Documents\NetBeansProjects\Ex1
☐ Use Dedicated Folder for Storing Libraries
Libraries Folder:
Different users and projects can share the same compilation libraries (see Help for details).
☑ Create Main Class: ex1.HelloWorld

The 'Project Name': Enter the name of this project "ex1" (or whatever you want to call your application).

Use a suitable 'Project Location' – the 'Project Folder' field underneath will follow the changes automatically. The Project Location should ideally be in your local network folder.

Ensure that the 'Create Main Class' is checked.

Change the name of the class name in the Create Main Class edit field. This is always in two parts – the project name and your class name, separated by a full stop. In this example, change the default from ex1.Main to ex1.HelloWorld.

NetBeans default class naming behaviour may be confusing! - it sets the primary class name to the project name.

Finally, click the Finish button.

Inspect code generated by NetBeans

The HelloWorld project should now be displayed in the Project window and the source code file HelloWorld.java opens in the Source editor. This is illustrated in the next screen shot. Notice that the source code file opened in NetBeans already has some of the preliminary work done for you. This is the window that has the tab HelloWorld.java.

Don't worry about the package statement for now, and the constructor (`public main () {}`) – these will be covered later this week.

Take some time to review the contents of the windows so you learn what to expect when you create your own applications.

Entering your source code

Where you see the line of comment `// TODO code application logic` here in the NetBeans generated file, replace it with:

```
    System.out.println("Hello World");
```

Notice the IDE (Integrated Development Environment) tries to help you by including both sets of double quotes " " for you as you enter your text.

Did you include the '`;`' semicolon? In the example above, the semicolon has been omitted. After a second or two, NetBeans, which has been quietly checking your code in the background, highlights erroneous lines of code by underlining them in red. If you 'hover' the mouse over the line, a tooltip pops up describing the error that NetBeans has detected (very useful when you're new to programming!)

Do you still have all of the { (open) and } (close) braces? There should be four of them. Also, check that every open parenthesis has matching close parentheses. Again, if there are any brackets missing, NetBeans will detect that and will highlight the corresponding bracket in red and/or place a (!) sign on the left margin.

Dr Tomasz Kurowski
t.j.kurowski@cranfield.ac.uk

Compiling your application

The next step is to 'compile' your program. One way to compile a program with NetBeans is to click the 'Build Main Project' icon (the hammer!) on the menu bar, or 'Clean and Build Main Project' (the hammer with the broom!). This is shown in the next screenshot. Alternatively, you can use the shortcut key F11, or the menu items 'Build' / 'Build Main Project'.

If you have still have errors, NetBeans will try to help you out some more. The error listing is in the bottom window labelled Output – ex1(jar). In this case, we have no errors. If you click on the error statement, the cursor in the main source code window should jump to the offending line.

Correct any errors and compile it again (Give me a shout if you're still struggling!). Note that NetBeans is automatically saving your files as they change. There is generally no need to specifically save your source code files.

Running the program

Now let's see if the Java program works. Click the 'Run Main Project' icon (The green play icon) on the menu bar. Alternatively, use the shortcut key F6, or the menu items 'Run' / 'Run Main Project'. Any output from you program should appear in the bottom window again.

Project files

As well as automatically saving your source files, the NetBeans application does several other things for you behind the scenes while you are working on the source code. For example, in the folder C:\JavaPracticals\ex1\dist you will find a file called ex1.jar. This jar file (java archive) contains all the executable byte code from your project, including any libraries you included. It can now be distributed to any other system and run there.

Whilst this might be no easier than sending the Hello.class file at the moment, it becomes important when an application is developed from a number of different files. The jar file greatly helps the deployment of large projects with multiple source files.

Look over in the Projects window and browse through the various files that were created.

Closing a project

Just select the 'File', 'Close' menu items to close your project, and 'File' 'Exit menu items to quit the NetBeans application. To re-open a recently closed project, follow the menu to your project and navigate File > Open Recent Project > Project Name.

Variables, Calculations, and Dialog Boxes

Let’s try to modify the automatically generated program. Add a variable of type String to your program, initialised to your name, for example:

```
String s = "Harry Potter";
```

Change the `System.out.println()` statement to output the message “Welcome to Java from “, followed by your newly added String variable.

Exercises

1) Write a Java program to convert degrees Celsius to degrees Fahrenheit. The formula is:

$$degF = (degC * 9 / 5) + 32$$

Include `degC` as a constant in your code and display the resulting `degF` as a real value.

2) Create a new version of the previous program to display the original degrees Celsius, the calculated degrees Fahrenheit and suitable text between them, e.g.:

   ```
   “21 degrees C is the same as 69.8 degrees F”
   ```

3) The following version of the Hello World program contains a number of errors. Some errors will be detected at compile-time and some when running the program. Correct the program:

```
        private class BadWorld {
             public void Main (string[] args)
          {System.out.println (Hello
          World!)
        }
   }
```

4) Assign a value to a variable (`int time = 3456;`). Now convert this time value (representing seconds) to hours, minutes, and seconds, and display the results to the console.

5) Write a simulation of the operation of a vending machine giving change. Assign values to a couple of variables `costOfItem` and `amountTendered`. The program should display the change to be returned, using the lowest number of coins in the form:

   ```
   Number of 20p coins in change = 2
   Number of 5p coins in change = 3
   ```

Hint – work in pence and use the `%` operator. Use 200p (£2), 100p (£1), 50p, 20p, 10p, 5p, and 1p coins.

6) Modify the degrees Celsius to degrees Fahrenheit program from the earlier exercise to accept a degrees Celsius value as an integer number from an input dialog box.
7) Modify the previous program to accept the degrees Celsius value as a floating point number from an input dialog box.
8) Repeat exercises 6-7, but show your output in a `showMessage` dialog box instead of the system console.

Solution for Exercise 5

Most programming problems can be solved in a number of different ways. Here I provide you with one solution but but do not be discouraged if your approach is different, as long as your logic and results are correct!

If you're new to programing, my first advice to you would be to always try to break the problem into several small tasks. You can write them down on a piece of paper, or even as comments within your code. You can test your program after each task, which will help you to spot any errors easily.

Task 1: Getting input from the user

In this exercise, you will be asking for two inputs from the user: the cost of the item and the amount paid, although to simplify things you could also make the cost of item a constant value within your code. Implement a user input dialog box for the cost of the item (in pence) as follows:

```
String inputCost = JOptionPane.showInputDialog(
      null,
      "Enter The cost of the item:",
      "Vendor Machine",
      JOptionPane.QUESTION_MESSAGE
);
```

Because the input from a dialog box has to be a string, we now need to convert it to an Integer using the `parseInt()` method as follows:

```
int itemCost = Integer.parseInt(inputCost);
```

Now you need another dialog box to get the amount paid (in pence):

```
String inputPaid = JOptionPane.showInputDialog(
      null,
      "Enter The cost of the item:",
      "Vendor Machine",
      JOptionPane.QUESTION_MESSAGE
);

//Parse the user input to an integer
int amountPaid = Integer.parseInt(inputPaid);
```

Task 2: Calculating the change in £2 coins and remaining change

First, you need to calculate the change given, in pence:

```
int change = amountPaid - itemCost;
```

Dr Tomasz Kurowski
t.j.kurowski@cranfield.ac.uk

Then, you should calculate the number of £2 coins owed as change. Since a £2 coin contains 200 pence, we need to divide the total change owed by 200. It's a good practice to cast the remainder value to Integer in case the outcome of this division is not an integer; this will also round the result down, so that the result will be zero for values smaller than 200.

```
int twoPounds = (Integer) change / 200;
int remainder = change % 200;
```

The `remainder` variable contains the remaining change, which will need to be given in coins of smaller denominations.

Test your application by printing the outcome so far in the command line:

```
System.out.println("Two pound coins:" + twoPounds);
System.out.println("Remaining change: " + remainder);
```

If everything works fine, you can delete the second print statement, as it is not needed anymore.

It also makes sense to simply overwrite the value of the `change` variable with the amount of change remaining after subtracting the £2 coins (you could have also simply assigned the result of the modulo operation to `change` to start with, without ever using a `remainder` variable).

```
change = remainder;
```

Task 3: The £1 coins

Now we need to repeat what we did in Task 2 for £1 coins. We need to know how many one-pound coins are in the remaining change:

```
int onePound = (Integer) change / 100;
remainder = change % 100;
```

Test your application by printing the values to the command line:

```
System.out.println("One pound coins:" + onePound);
System.out.println("Remaining change: " + remainder);
change = remainder;
```

Repeat this for the rest of the coins!