# Introduction to BlueJ

The purpose of this lab is to allow you to become more familiar with the BlueJ Integrated Development Environment, (IDE). You should have already briefly used BlueJ in your first lectoral class.

## Preliminaries

Once you are logged on to the system, go to the Start menu, then Computing, then Java and then BlueJ. Click on the BlueJ option. This should start the BlueJ. You will see a window similar to the following:

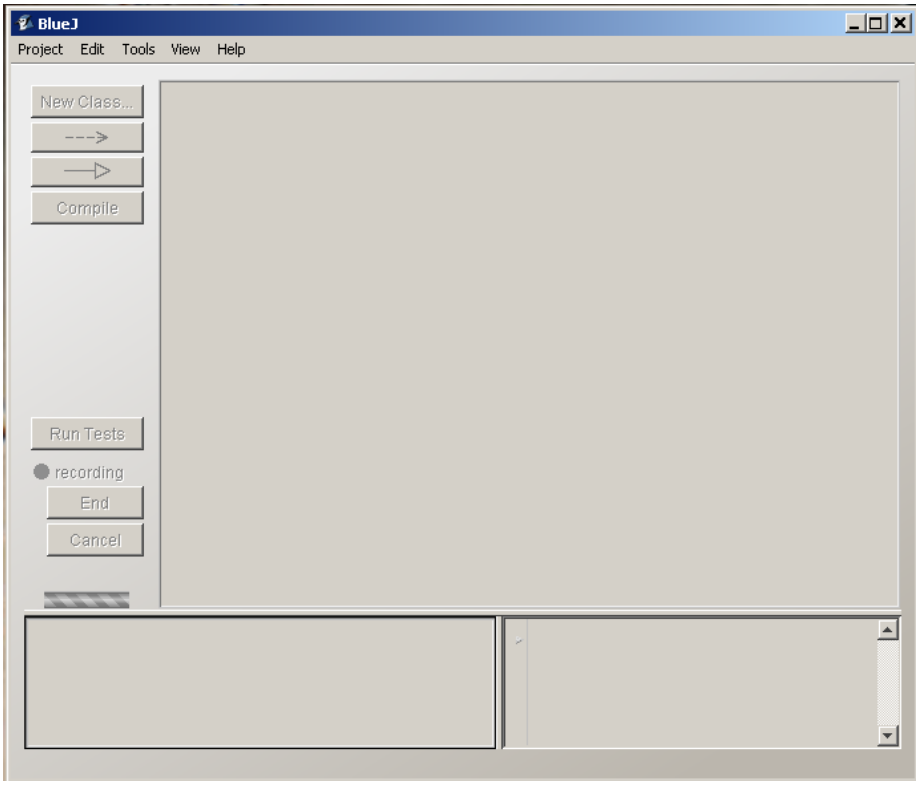


Figure 1: BlueJ IDE

## Starting to use BlueJ

In the Java world a program under development is known as a project. A Java program will typically consist of several files. The first thing you need to do is to set up your own project.

### Task 1

In the BlueJ window select “Project”-> “New Project”. (see figure 2)

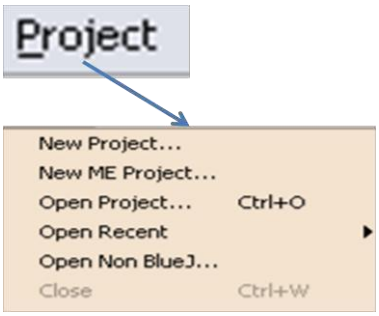


Figure 2:

You will be presented with a dialog window:

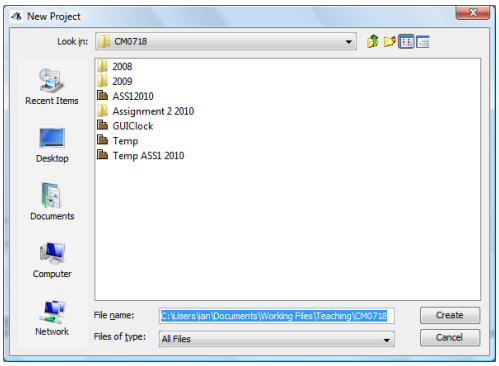


**Figure 3:**

Navigate to the location you want to use to for the project. This should be on your U: drive, probably in a folder called **SCC110\week01** that you created in the lectoral. Once there, enter the name of the project in the file name field. For this exercise we want you to call the project, **Shapes**.

Enter **Shapes** and press return. You have now created your first project. You have created a folder (Shapes) which contains two files:


The first is a text file that can be used to record notes about the project.
The next one belongs to BlueJ. It contain administrative information that is used by BlueJ. You cannot read it. Check that the folder ad files have been created using Windows Explorer.

## Task 2

The next stage is to create some files. Normally you would do this but for this part of the lab we are going to provide you will some files. You will find these on Blackboard. The files are called:

- **Canvas.java**
- **Circle.java**
- **Square.java**
- **Triangle.java**

Copy these files into you folder **Shapes**.

## Task 3

Select Edit and then select "Add Class from File". You will be presented with a file selection dialog. Move to the folder Shapes and select the 4 java files.

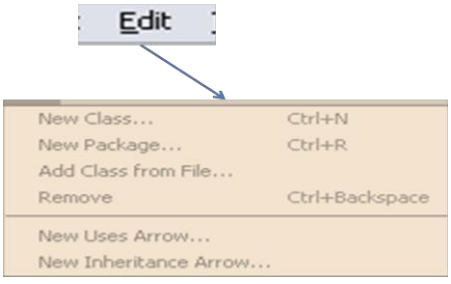


**Figure 4:**

You will need to click the Compile button in the BlueJ tool bar. You should now see something similar to this:

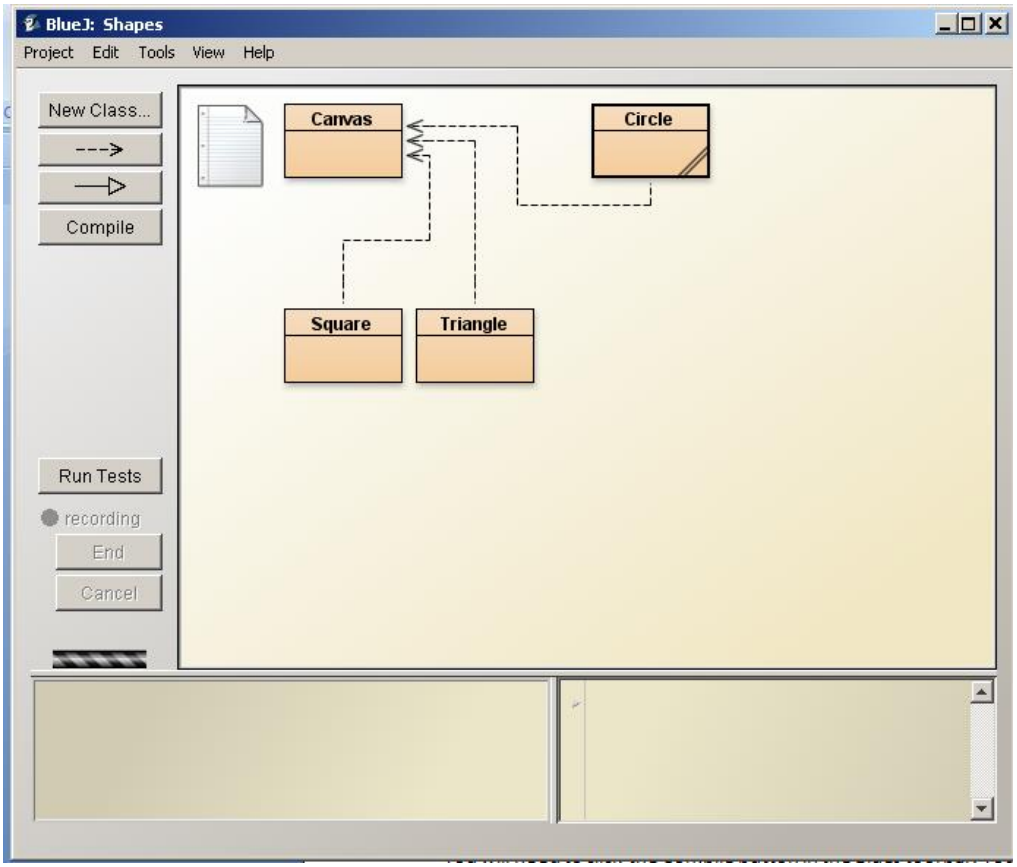


Figure 5:

## Task 4

You can now explore this project.

1. Right click on the Square class and choose new Square(). Click Ok in the 'Create Object' dialog that appears.
2. Right click on the Square object in the Object Bench.


You will see a list of the methods available for your use. Select makeVisible(). A small window will appear showing a square in its default position and default colour.

3. Experiment with **moveLeft(), moverRight(), moverUp(), moveDown() ,makeInvisible() and makeVisible()**.

## Task 5

If you invoke **moveHorizontal(int distance)** you will see the following dialog:

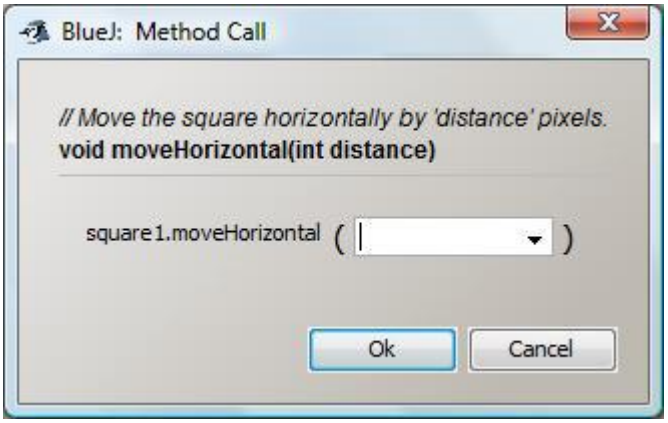


Figure 6:

You are being asked to enter some data. (You are being asked to provide a value for a ***parameter***.) In this case this will be an integer which equals the number of pixels you want to move the square – a positive number moves it to the right and a negative number to the left. Enter a value in the range 50 to 100.

Experiment with the other methods that require an integer parameter. There are four others.

## Task 6

There is just one method left to use – changeColor( String newColor). Invoke this method and you should see:


Figure 7:

**NOTE: Java is case-sensitive.**

**e.g. “red” not “RED”.**

The parameter this time is of type String. We will look at type String in the lectorals. For the moment take it to be a sequence of characters inside “ “ (double quotes). The acceptable strings are shown in the dialog box. Type one in.

## Task 7

Now create some circle and triangle objects and perform similar experiments with them.

## Task 8

You are going to start a new exercise using the classes in the shapes project and we need to reset the canvas. Go to Tools in the menu bar and select "Reset Java Virtual Machine". This has the effect of restarting the project.

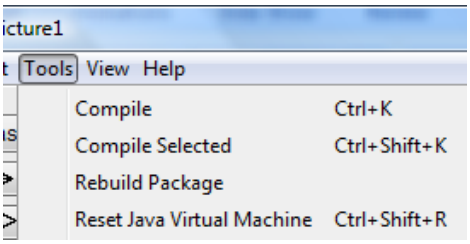


Figure 8:

Look at the following picture:

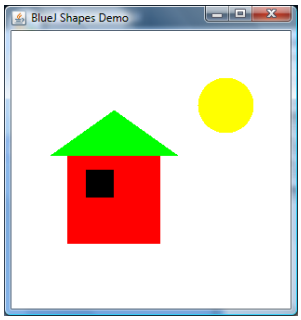


Figure 9:

By creating various objects and using their methods build this picture. As you do write down the steps you are taking. These steps are an "algorithm" for producing the picture. They will be useful for Task 9.

## Task 9

You are going to carry out some simple editing on a Java class that will, hopefully, draw the picture shown in figure 9. Close your current project and create a new one called Picture2. Add the four java files you used in the Shapes to this new project. (If you have forgotten how, look at Task 3 above).

From Moodle add the file Picture.java to the project. ( you will need to copy it into the Picture2 folder and then add it to the project). Once you have added it to the project double click on the Picture class or right-click and select "OpenEditor".

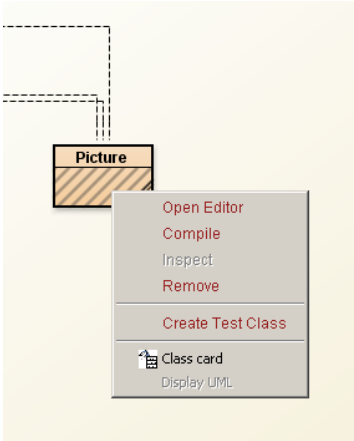


Figure 10:

Find the draw method. It contains four sections of code – two for squares, one triangle and a circle. Where ever you see the phrase "ENTER AN INTEGER VALUE" replace it with a suitable integer value and when you see the phrase "ENTER A COLOUR STRING" enter a suitable colour string, e.g. "red".

When you have entered all the necessary integer values and strings click **Compile**. You will get a message at the bottom of the edit window. Hopefully it will say "Class compiled – no syntax errors". If it doesn't say this see if you can understand what the problem is and then ask the lab tutor for help.

Once the file has compiled create a picture object and select the draw method for that object. Did you draw the correct picture? If not do you understand why not?

## Task 10

In the source code of class Picture, find the part that actually draws the picture. Change it so that the sun will be blue rather than yellow.

## Task 11

Add a second sun in the source code of class Picture. To do this, pay attention to the field definitions close to the top of the class. You will find this code:

```
private Square wall;
private Square window;
private Triangle roof;
private Circle sun;
```

You need to add a line here for the second sun. For example:

```
private Circle sun2;
```

Then write the appropriate code for creating the second sun.

## Challenge exercise I

(This means that this exercise might not be solved quickly. We do not expect everyone to be able to solve this at the moment. If you do – great. If you don't, then don't worry. Things will become clearer as you read on. Come back to this exercise later.)

Add a sunset to the single-sun version of Picture. That is: make the sun go down slowly. Remember: The circle has a method slowMoveVertical that you can use to do this.

## Challenge exercise II

If you added your sunset to the end of the draw method (so that the sun goes down automatically when the picture is drawn), change this now. We now want the sunset in a separate method, so that we can call draw and see the picture with the sun up, and then call sunset (a separate method!) to make the sun go down.