

# INTPROG TB2: OBJECT ORIENTED PROGRAMMING

## Practical Worksheet 3: Introduction to the BlueJ Environment

### 1. Introduction

The aim of this practical session is familiarise yourself with the BlueJ environment and how we can create classes using Java. Work through this practical sheet at your own pace, asking questions if you need clarification. Next week, we will start to go into more detail relating to developing Java classes. However, for this week, concentrate on getting used to the BlueJ IDE.

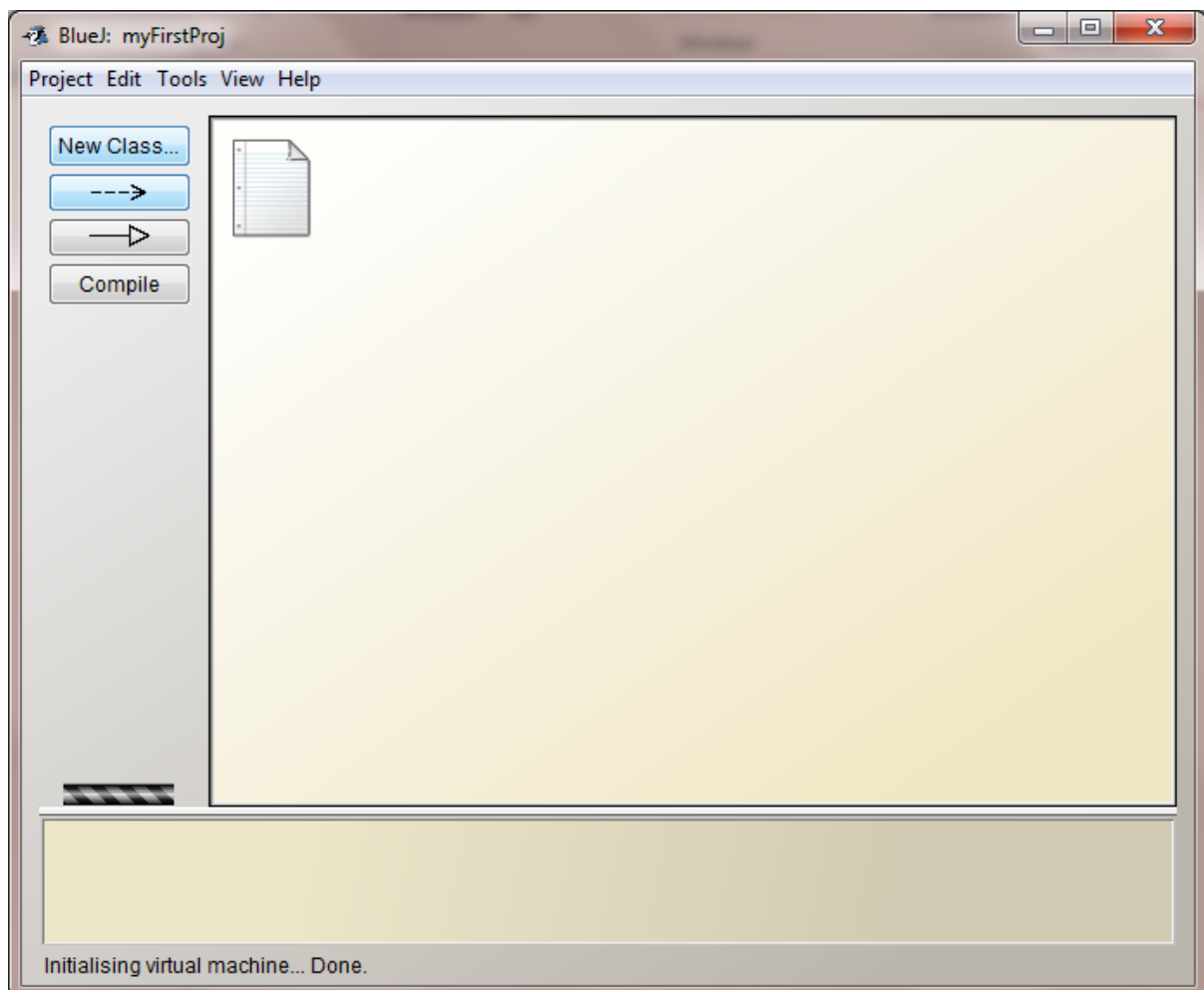
### 2. Creating your First Class in BlueJ

To get used to coding in Java, and using objects, we will be using the BlueJ IDE...

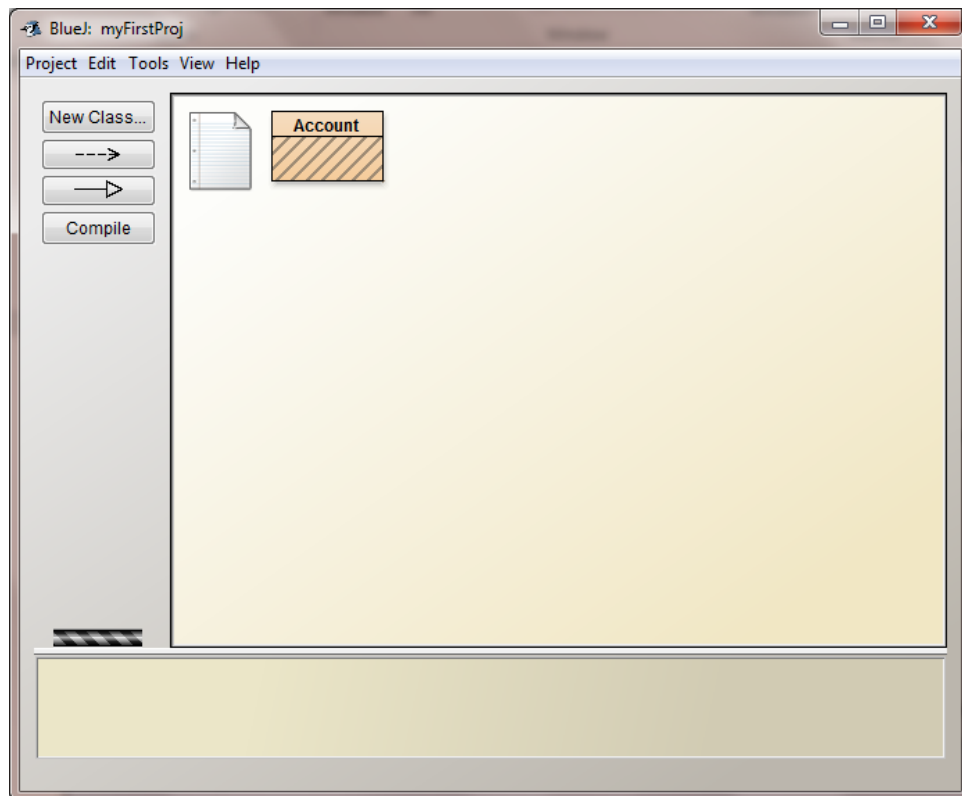
For the rest of this teaching block, we will be using either the BlueJ or Netbeans IDEs. The focus for this week will be BlueJ. Once we start getting more confident in using Java, we will start to use Netbeans.

BlueJ is available on all computers at the University through AppsAnywhere. It is, however, free to download on all major operating systems (Windows, MacOS, Ubuntu/Debian, Raspbian) from <http://www.bluej.org>. There is also support for 'other' systems using the following instructions: <http://www.bluej.org/generic-installation-instructions.html>.

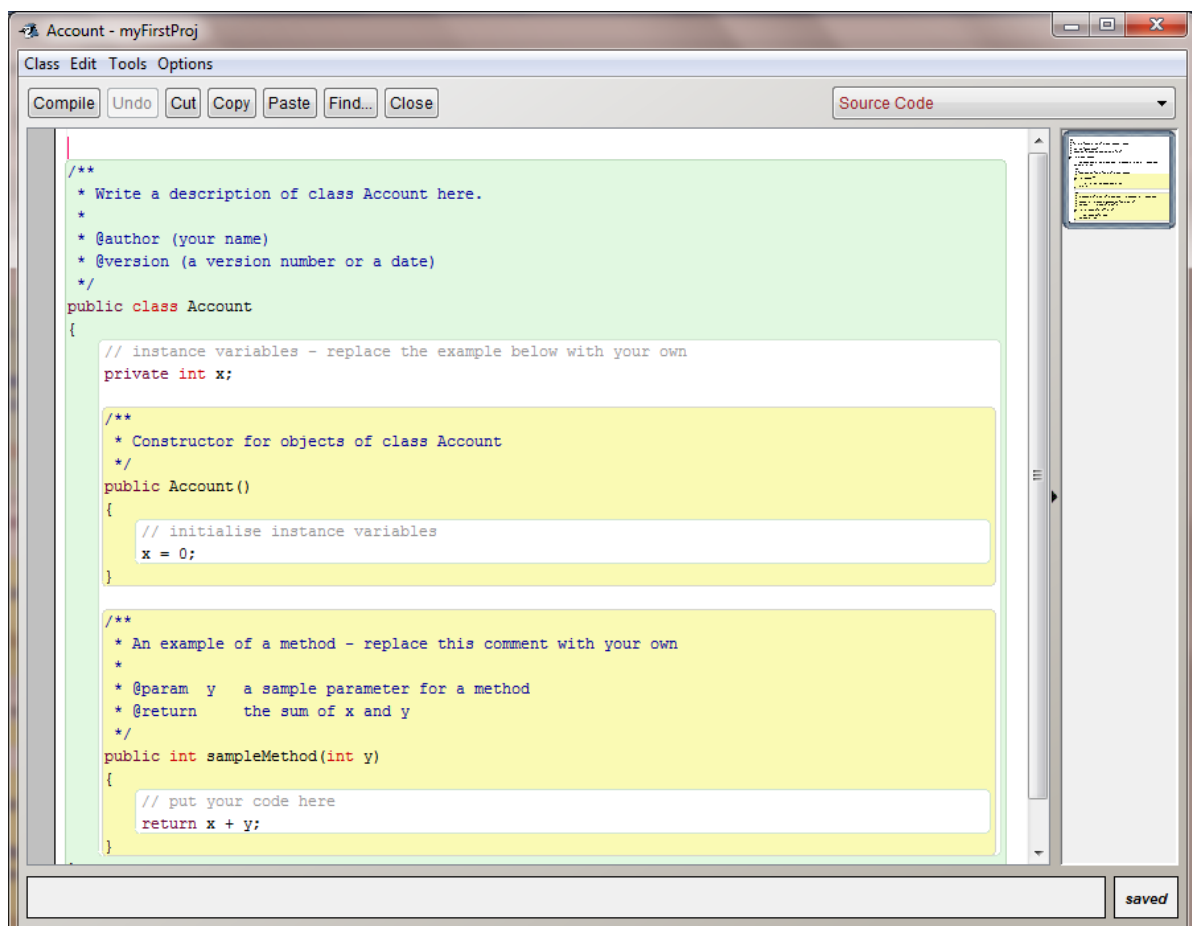
Start by opening up BlueJ. Once it has opened, select New Project from the Project menu. Call your new project 'myFirstProj' and save it somewhere appropriate on your N: drive.



Click the New Class button, calling it 'Account'. Make sure that the Class Type is 'Class'. Click OK. Your class should be created in the window.



Right click the Account block, and select Open Editor.



Highlight all the code, then delete it.

Enter the following code. The comments (which are preceded by //) are there to explain what the code is doing. You do not need to copy these across, unless they will be of benefit to you when looking back at your code.

```
// declaration of the class - the name must match the name of the class when you created it
public class Account
{
    // Declaration of the instance variables to be used within the class
    private String accountName;
    private int accountNumber;
    private int currentBalance;
    private int overdraftLimit;

    // Constructor
    // The name of the constructor must be the same as the class
    public Account(String name, int number, int overdraft)
    {
        accountName = name;
        accountNumber = number;
        overdraftLimit = overdraft;
        currentBalance = 0; // balance will be 0 at creation
    }

    // Method to deposit money
    public void deposit(int anAmount)
    {
        currentBalance += anAmount;
    }

    // Method to withdraw money
    public int withdraw(int anAmount)
    {
        // variable which will only be used in this method
        int totalAvailable = overdraftLimit + currentBalance;

        if (totalAvailable > anAmount) // person has enough money
        {
            currentBalance -= anAmount;
            System.out.println("Withdrawal was successful");
        }
        else // person doesn't have enough money
        {
            System.out.println("Insufficient funds");
        }

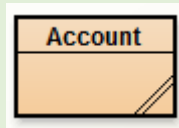
        return currentBalance;
    }

    // Method to print account information
    public void printInformation()
    {
        int availableFunds = overdraftLimit + currentBalance;

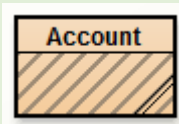
        System.out.println("Name: " + accountName);
        System.out.println("Account number: " + accountNumber);
        System.out.println("Current Balance: " + currentBalance);
        System.out.println("Overdraft Limit: " + overdraftLimit);
        System.out.println("Available Funds: " + availableFunds);
    }
}
```

Click the compile button at the top of the screen. If you have entered the code correctly, it should not produce any errors. Close the editor once you have been able to compile the class without any syntax errors.

Note: A correctly compiled class will be a solid orange colour.



If the class has diagonal lines across it, the compile process was not successful – you need to check your class does not have any errors. A message with syntax errors will be displayed at the bottom of the screen.

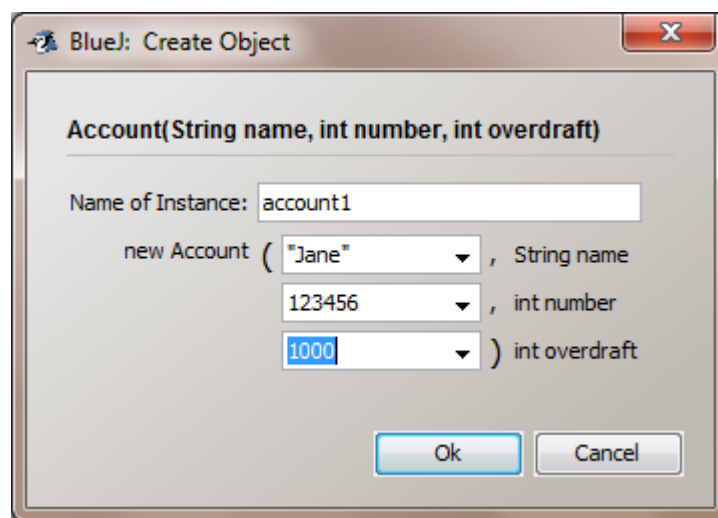


Only classes which have been compiled can be run – each time you change a file, you will need to recompile the file.

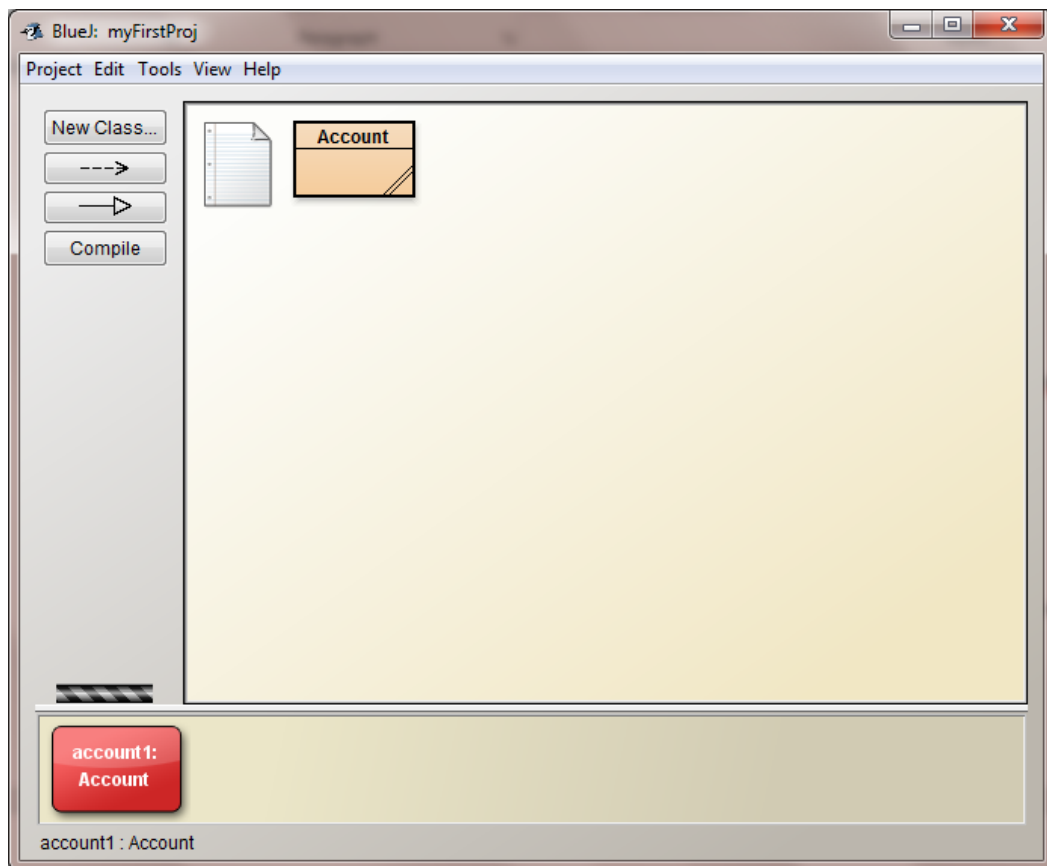
### 3. Creating Objects in BlueJ

Now that we have our class, we can create multiple instances of it - essentially creating objects.

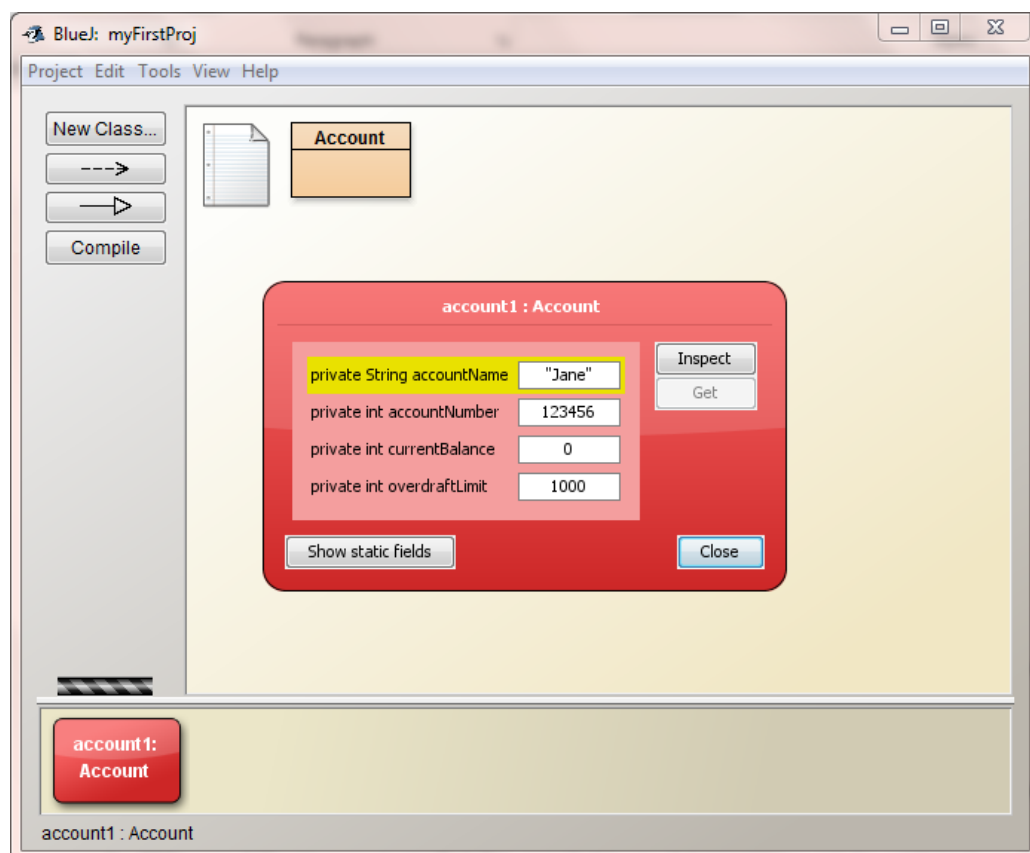
Right click the class, and select `new Account(String name, int number, int overdraft)`. Call the name of the instance `account1`, and enter "Jane", 123456, and 1000 between the two brackets.



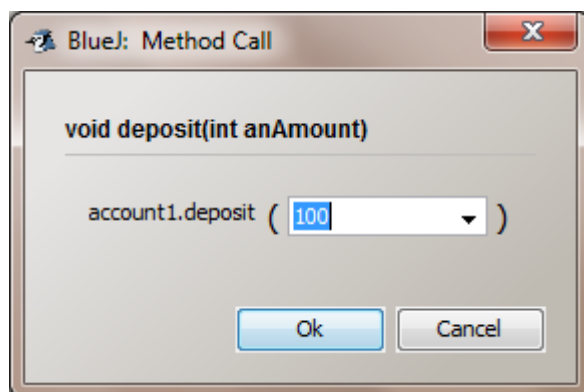
This will construct your object, with the `accountName` set as "Jane", `accountNumber` set as 123456, and `overdraftLimit` set as 1000. This object will be displayed at the bottom of the screen in a red block.



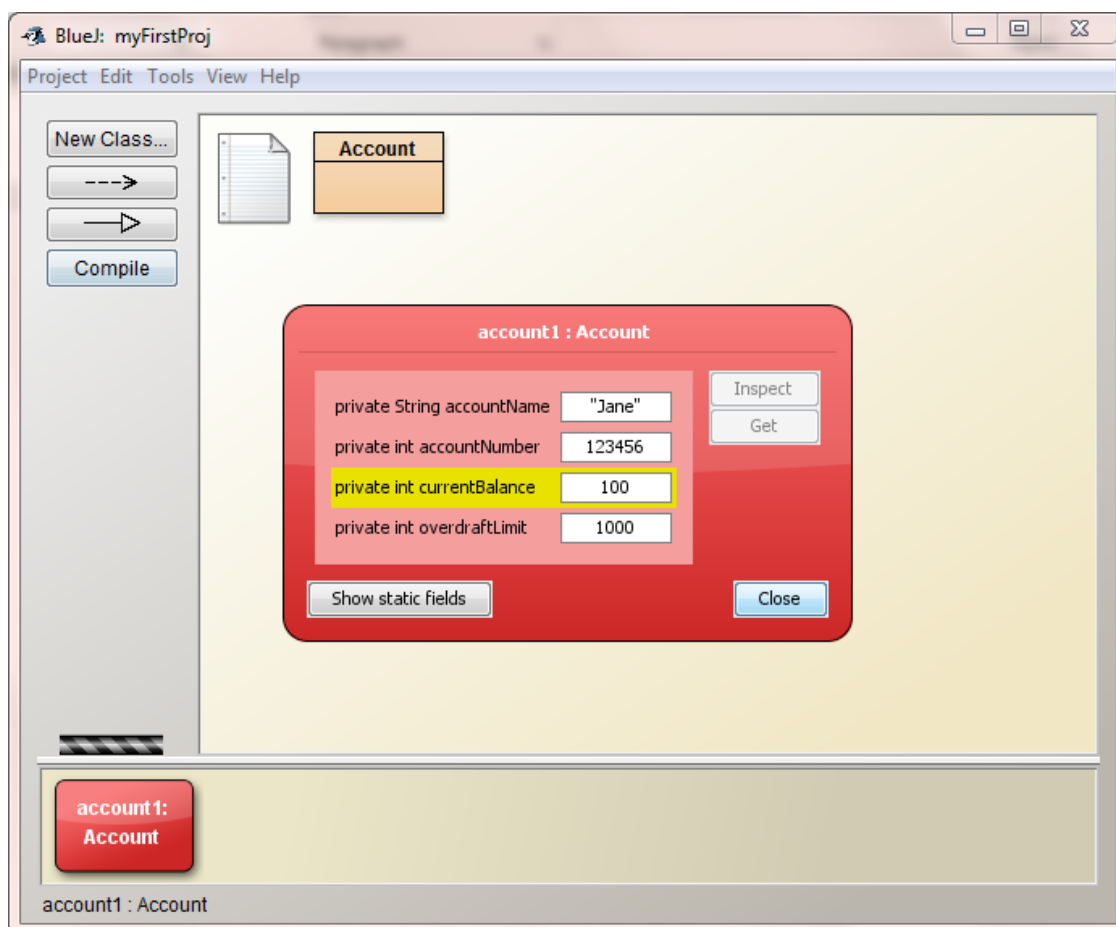
By double clicking on the object, we can inspect the value of each variable within the object.



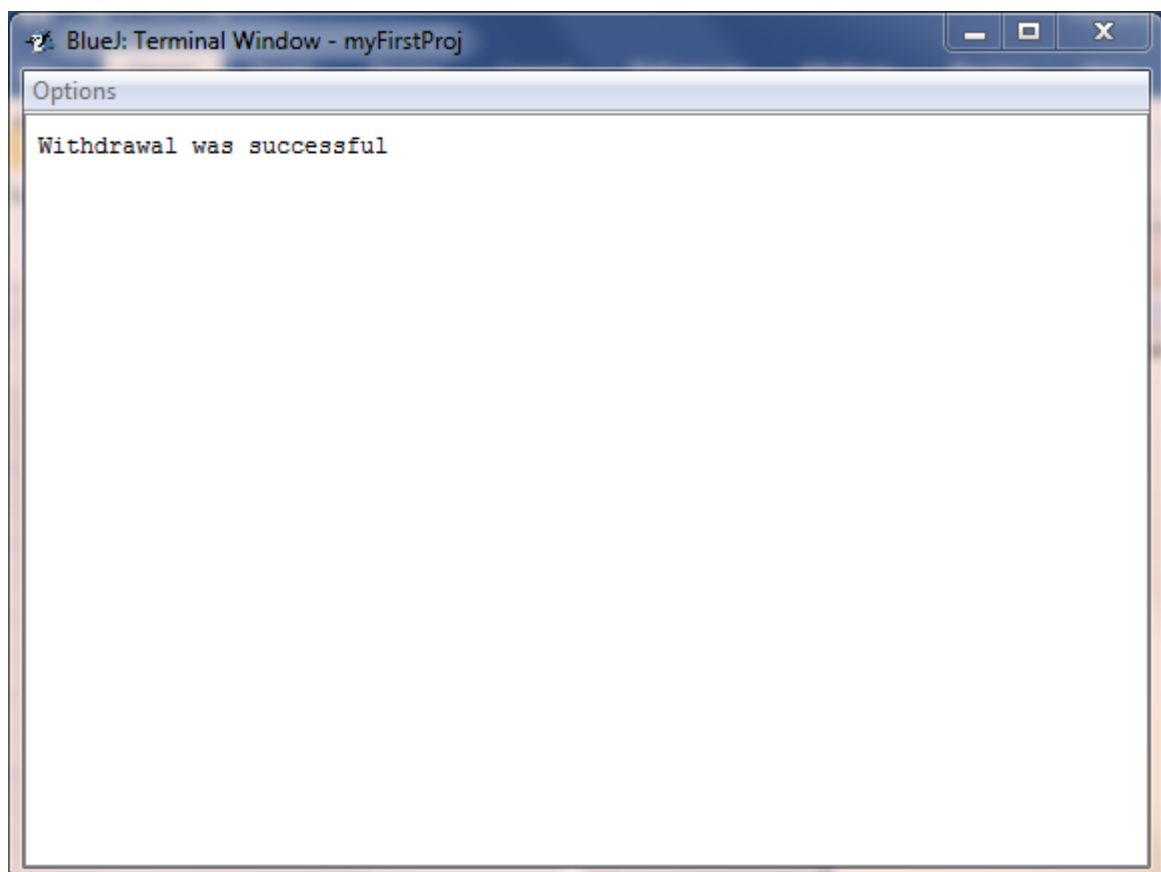
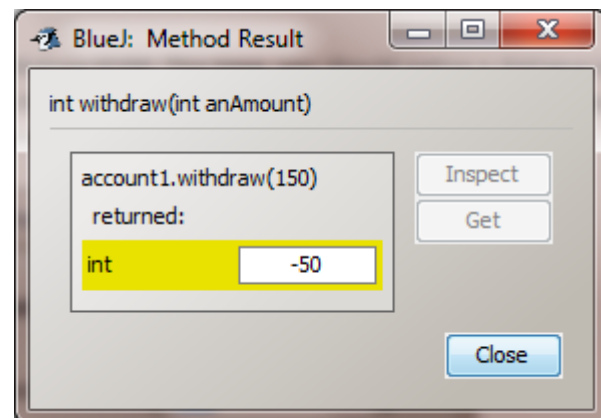
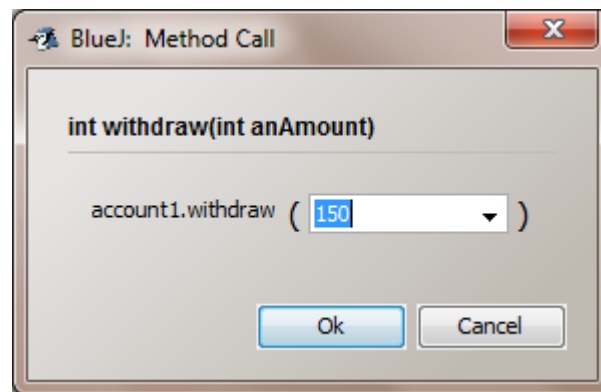
Right-click the account1 object, and select the void deposit(int anAmount) method. Enter 100 and click OK to deposit £100 into the account.



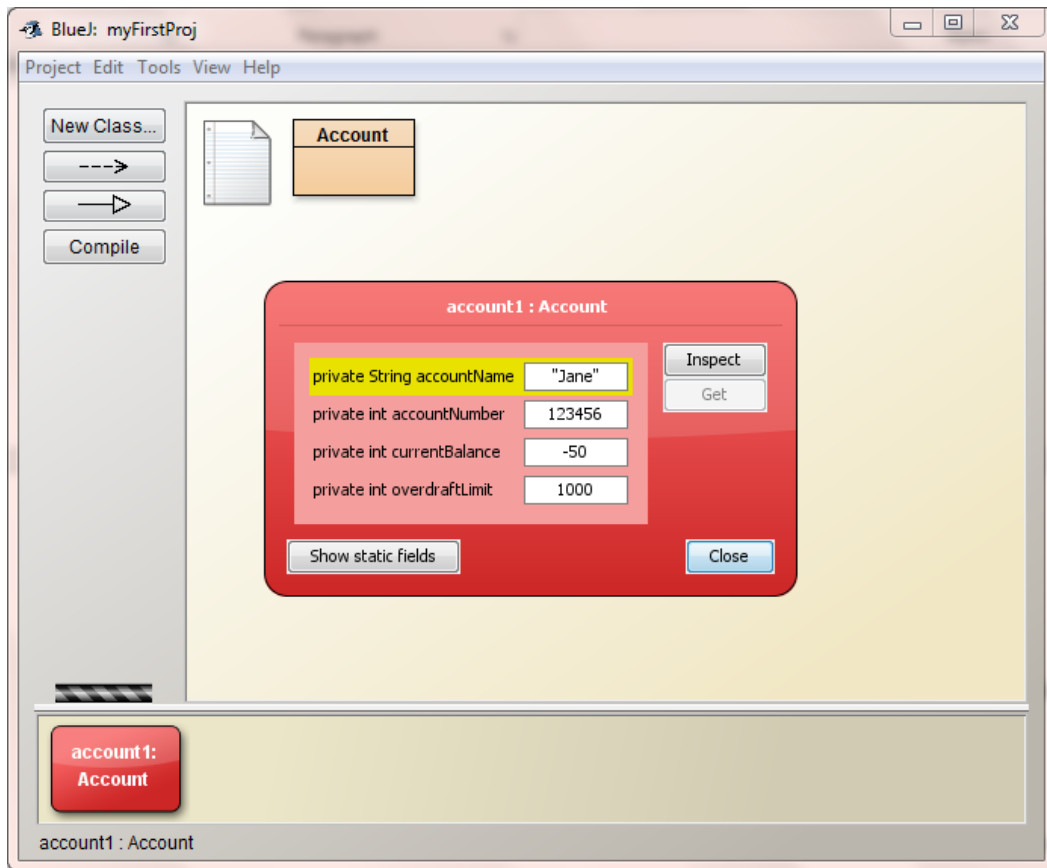
Double click the object to inspect the value of the variables in the account1 object.



Now run the `int withdraw(int anAmount)` and withdraw £150. This time, you should notice a dialogue box appears to show the result which has been returned by the method (in this case, the amount left in the account after the money has been withdrawn). You will also notice that a terminal window has opened containing the print statement.

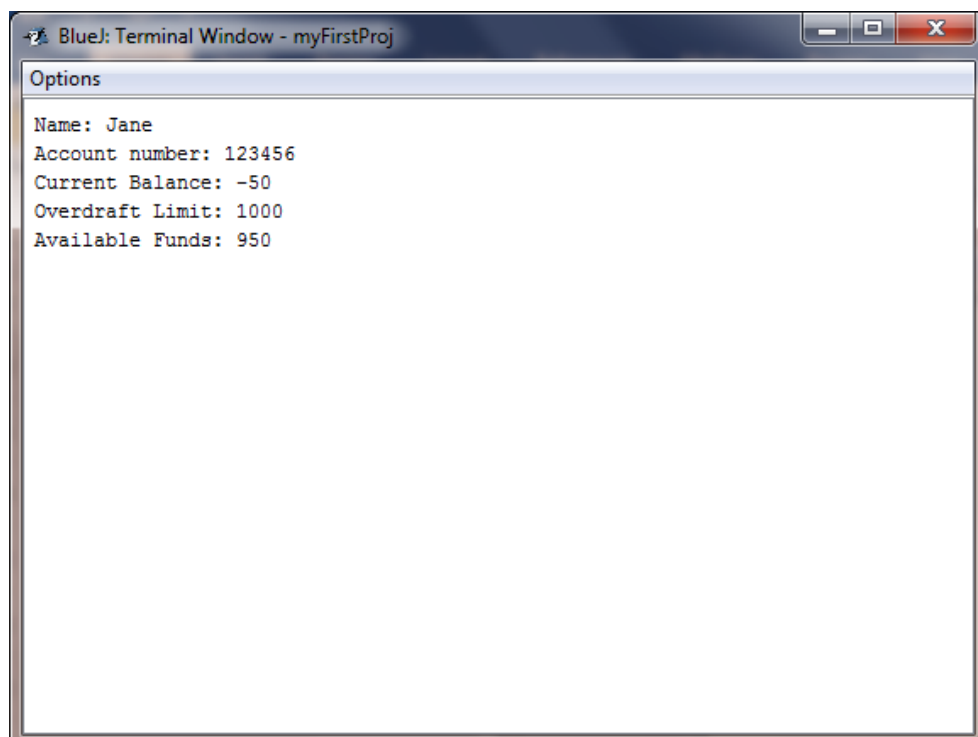


Once again, inspect the value of the variables in the account1 object.



Create another object, with the name set to Fred, the account number 987654, and an overdraft of 0. Fred deposits £200 into the account, then withdraws £125. At each point, inspect the values stored within the object.

Finally, run the `printInformation()` method for `account1`. You should get the following output in the terminal window:





## 4. Creating and Using Objects in Other Files

This section will create a Test class, which will create Account objects, and run their methods.

Click the "New Class" button, and name the new class "Test".

Open the editor for the Test class, and replace the boilerplate code with the following:

```
public class Test
{
    public static void main(String[] args)
    {
        // declare a variable of type Account
        Account account_1;

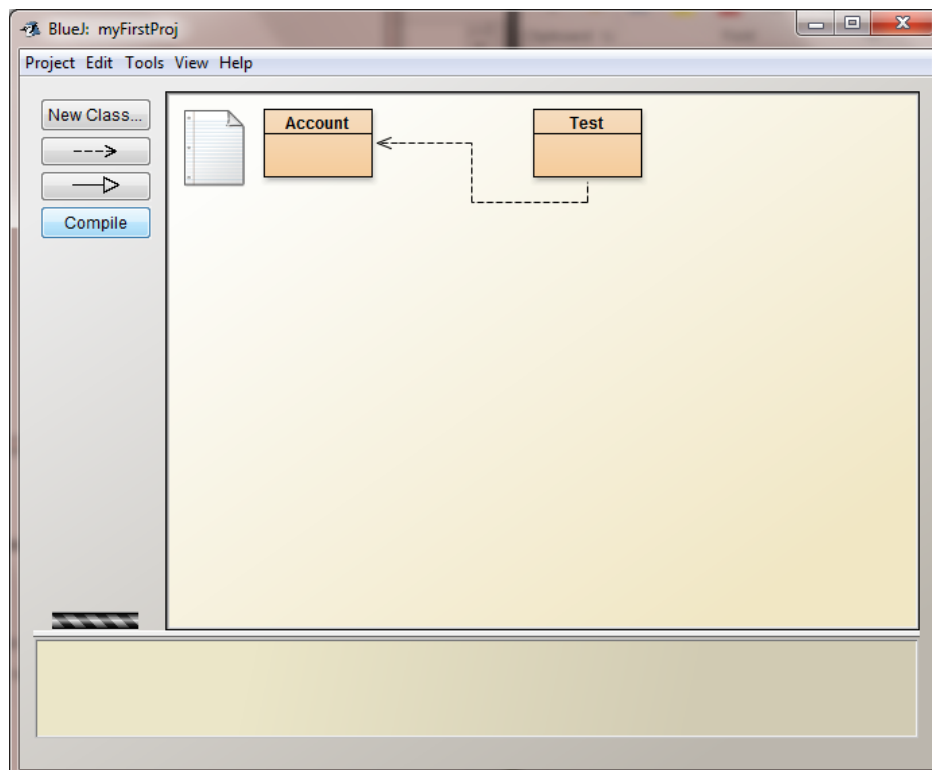
        // create an account object with required details
        account_1 = new Account("Jane", 123456, 1000);

        // run methods contained within class
        account_1.printInformation();
        System.out.println();

        account_1.deposit(100);
        account_1.printInformation();
        System.out.println();

        account_1.withdraw(150);
        account_1.printInformation();
    }
}
```

Click the compile button, and once there are no errors, close the editor.

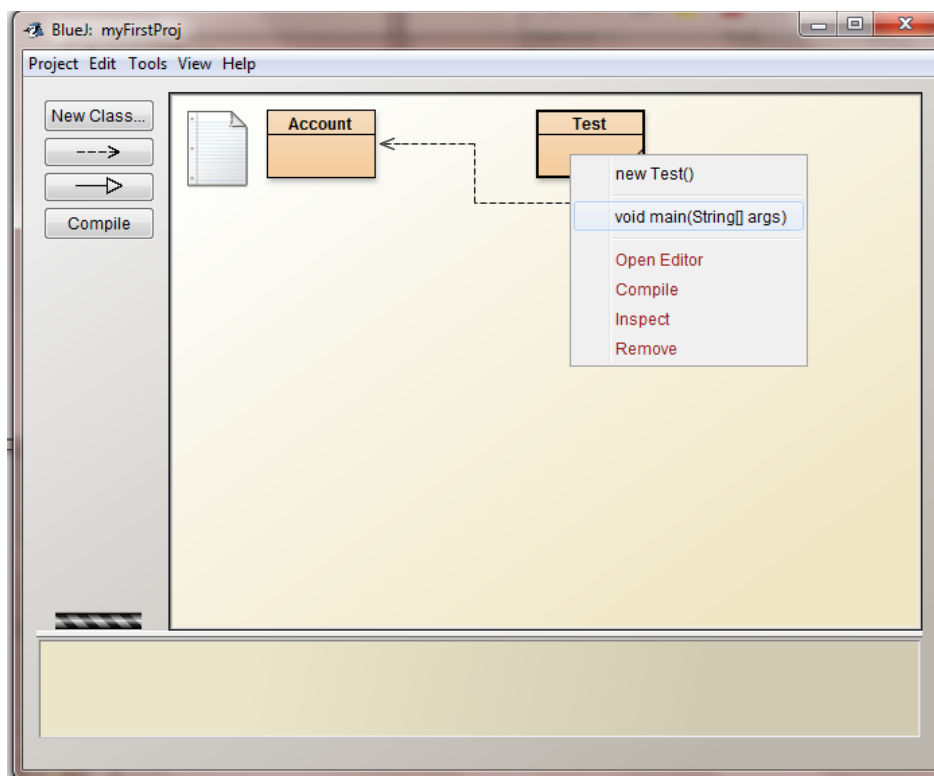


In this part of the example, we are going to run the method directly rather than creating an object. In Python, you defined a `main()` function. In Java we can do something similar, which is defined using the following code:

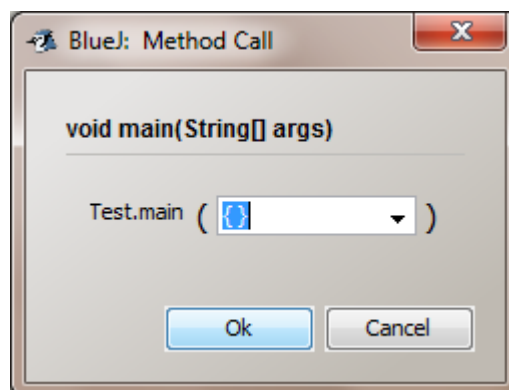
```
public static void main(String[] args)
{
    // code here
}
```

The main method has been used in this example. It always has the same definition. The keyword `static` means that the method belongs to the class rather than the object. It means that we are able to run the method without creating an object. Other than in the main method, we will not be using the `static` keyword. Therefore, we will not go into detail about the `static` keyword.

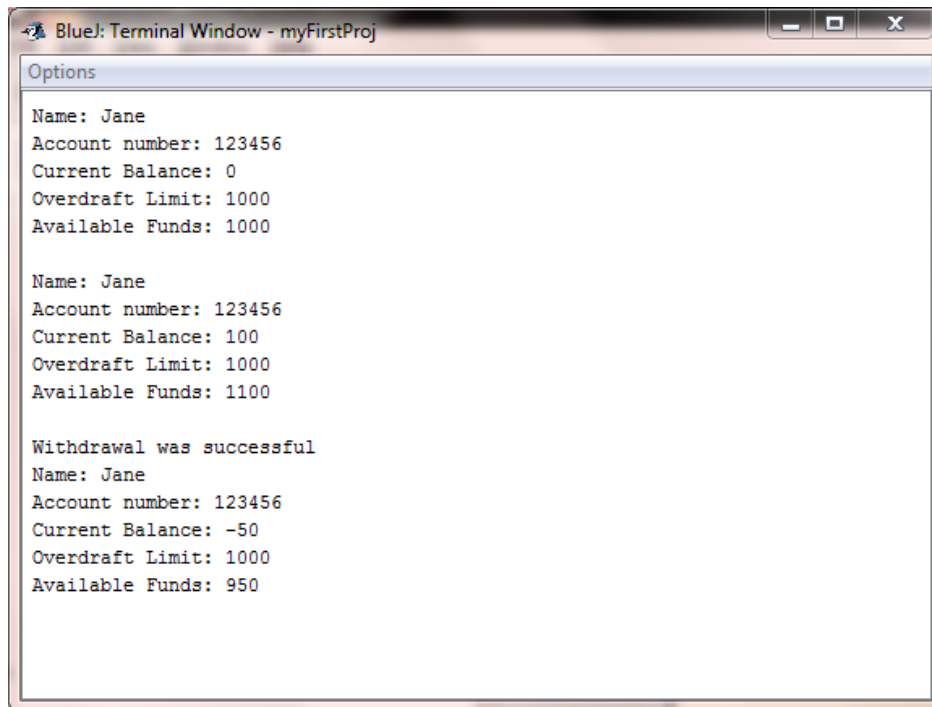
To run the main method within the class, right-click the Test block. Click on `void main(String[] args)` in the pop up box.



When the Method Call box appears, just click OK to keep the default argument.



This will not create an object of type Test, but will run the main method. The following will be output in the terminal window:



```
Options
Name: Jane
Account number: 123456
Current Balance: 0
Overdraft Limit: 1000
Available Funds: 1000

Name: Jane
Account number: 123456
Current Balance: 100
Overdraft Limit: 1000
Available Funds: 1100

Withdrawal was successful
Name: Jane
Account number: 123456
Current Balance: -50
Overdraft Limit: 1000
Available Funds: 950
```

## 5. Practice Questions

These questions are designed to get you to explore the BlueJ environment – we will start building more classes and using them next week.

1. Make some deliberate errors in your Test class, compile it to see what messages you get. You are trying to see the effect of single errors, so make sure you revert back to the working class before making another deliberate mistake. I would recommend taking note of the error message – then when you get it in the future, you will know what is causing it. Can you understand the error messages? How can you get more information about the error from BlueJ?

Some suggested errors:

- Delete ; from the end of a line in the Test class
  - Mistype "void" to "Void" in the Test class
  - Misspell "new " as "now" in the Test class
  - Misspell "Account" as "Accont" in the Test class
  - Misspell "public" as "pablic" in the Test class
  - Replace "=" with "==" in the Account class
  - Delete the line "Account account1" in the Test class
  - Delete the last "}" in the Account class
  - Delete the first "{" in the Account class
  - Replace the "." with " " in one line which is account\_1.printInformation() in the Test class
2. Find out how you can display line numbers in the editor
  3. The editor displays the source code of a class. How do you display the documentation of a class? What is in the documentation?
  4. Without leaving BlueJ, find the BlueJ tutorial and have a look at what is in it.
  5. Modify the Test file so instead of creating an account owned by "Jane" it is owned by "Fred".
  6. Modify the Test file so £200 is deposited and £125 is withdrawn from the account.
  7. Create a new account in the Test file which is owned by Alice, has account number of 245819, and has no overdraft. Add lines to the Test file so that £1000 is deposited, and then withdraw £1200.