Programming In Java

Object Orientated Programming

Niall Curley

BMC Consulting  Millfield Belfast

Contents

[Introduction 3](#_Toc482363449)

[Java Programming 3](#_Toc482363450)

[History 3](#_Toc482363451)

[The Structure of a Java Program 3](#_Toc482363452)

[A class 3](#_Toc482363453)

[ Access Modifier 4](#_Toc482363454)

[ Return Type 4](#_Toc482363455)

[ Method Name 4](#_Toc482363456)

[ Parameter List 4](#_Toc482363457)

[ Method Body 4](#_Toc482363458)

[Void 5](#_Toc482363459)

[Java Language 6](#_Toc482363460)

[Cross Platform 6](#_Toc482363461)

[Keywords 7](#_Toc482363462)

[Variables 7](#_Toc482363463)

[Numeric (integers) 7](#_Toc482363464)

[Numeric (Real Number) 7](#_Toc482363465)

[Boolean 7](#_Toc482363466)

[Textual 7](#_Toc482363467)

[Literal 7](#_Toc482363468)

[Declaring Variables 7](#_Toc482363469)

[Identifiers 8](#_Toc482363470)

[Classes 8](#_Toc482363471)

[Methods 8](#_Toc482363472)

[Constants 8](#_Toc482363473)

[Operators 8](#_Toc482363474)

[Example of Arithmetic Operations 8](#_Toc482363475)

[Input from the Keyboard 9](#_Toc482363476)

[Commenting 9](#_Toc482363477)

[Indentation 9](#_Toc482363478)

[Import Classes & Libraries 10](#_Toc482363479)

[Selection in Java 11](#_Toc482363480)

[Nested if else Statements 11](#_Toc482363481)

[switch 12](#_Toc482363482)

[Iteration in Java 12](#_Toc482363483)

[The While Loop 12](#_Toc482363484)

[Do While Loop 13](#_Toc482363485)

[The for Loop 14](#_Toc482363486)

[Object Orientated Java (OOP) 15](#_Toc482363487)

[Classes 15](#_Toc482363488)

[Programme class 15](#_Toc482363489)

[Library class 15](#_Toc482363490)

[The Object class 15](#_Toc482363491)

[Properties 15](#_Toc482363492)

[Methods & Types 15](#_Toc482363493)

[Object Class & Programme Class 16](#_Toc482363494)

[Instance 16](#_Toc482363495)

[Calling the method and dot operator 17](#_Toc482363496)

[Inheritance 17](#_Toc482363497)

[Encapsulation 18](#_Toc482363498)

[Overloading/Overriding 18](#_Toc482363499)

[Method Overriding 19](#_Toc482363500)

[Evaluate CLI with IDE 20](#_Toc482363501)

[CLI 20](#_Toc482363502)

[Compiler 20](#_Toc482363503)

[Javac 21](#_Toc482363504)

[Integrated Development Environment (IDE) 22](#_Toc482363505)

[IDE Colour Coding 23](#_Toc482363506)

[Run & Output 23](#_Toc482363507)

[Top Menu 24](#_Toc482363508)

[Java Libraries 26](#_Toc482363509)

[Main Differences Advantages & Disadvantages 26](#_Toc482363510)

[Bibliography 28](#_Toc482363511)

# Introduction

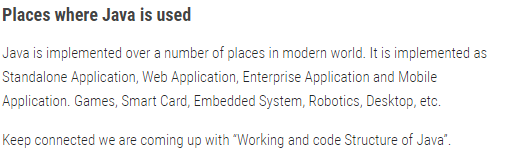
As part of my role in BMC Consulting I have been asked to write a technical report on Java programming and Java Development using different environments. It will contain technical information on Java programming and Object Orientating Programming. It will make a comparison between the uses of a Command Language Interface against that of an Integrated Development Environment.

# Java Programming

## History

Java was created in 1991 by a team of computer scientists (Green Team) led by James Gosling. It was developed to make computers part of our daily lives rather than just part of computer networks. Today Java is behind most of the application software we use on our mobile technology, games and ecommerce.

According to techmit;[[1]](#footnote-1)



## The Structure of a Java Program

The most basic Java program consists of something called a ‘class’ and a ‘main method’, this is also known as the programme class, as only the main method can run the Java programme.

A class - is an object that has a common structure and behaviour to other objects. A class is given a name (a class identifier) by the developer according to a certain convention. This will be of mixed case with the first letter of each word in capitals. A should be named appropriately to indicate what it will be used for, e.g;

public class ShopAccount {

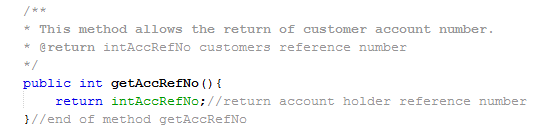
The keyword ‘**public’** is called an ***access modifier*,** ‘public’ means that the coding used in this class can be accessed by other object classes.

The next layer of code, **‘public static void main (String [] args)’** is the main method. **A method** is group of statements that together perform an operation.

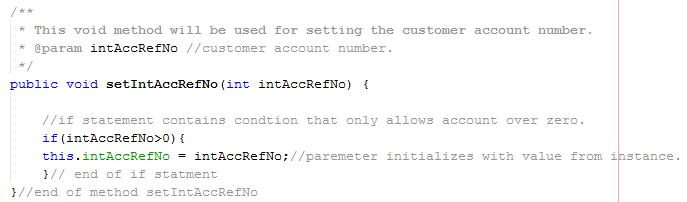
A method has a;

* Access Modifier – which defines the type of access, e.g ‘public’
* Return Type – determines what value should or should not be returned e.g **String, double, int.** Or **void** which means it has no return value.
* Method Name – The name given to the method and optionally a parameter list. e.g **setAccount ();**
* Parameter List – this is a list of local variables that are used in conjunction with the creation of an instance, setAccount(intNewRef);
* Method Body – This is the curly brackets that enclose any statement made in the programme. E.g setAddress() {//block for actions to be carried out}

#### Example of a Method



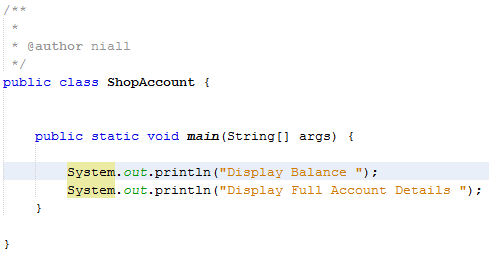
#### Example of method with parameter



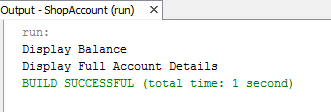
## Void

**Void** means that the method does not return a value when the **method is called. A method can give an answer or not.**

The***instructions*** (statements) for the programme will be within the main method as below. So ‘System.out.println (“Display Balance”) is the instruction for the programme to be outputted, that is displayed on the screen e.g ‘**Display Balance’**.



Curly brackets are used after the opening of the class and the closing of the class. They are also used at the start of the main method and they enclosed the instructions as in above.

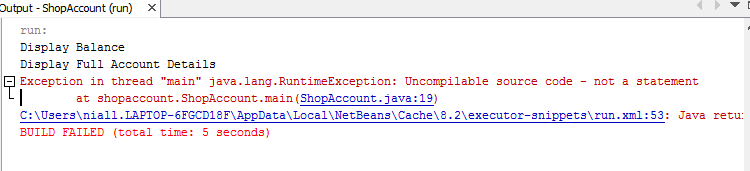


The output is displayed when the programme is run (as above).

# Java Language

When the Java programme is run, it is compiled by the Java Virtual Machine, it interprets compiled Java binary code (bytecode) for the processor so that it can carry out Java instructions. You will see when you compile a programme that .class file is created that has the same name as the .java file, this file contains the byte code which the VRM uses.

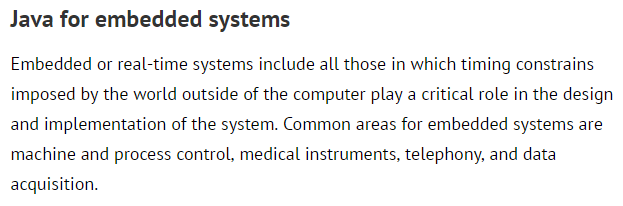
The language of Java is structured into several different aspects that the compiler will recognise. If there are any discrepancies in the code, it will usually not run and report ***an error***.

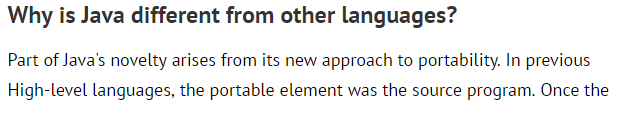


## Cross Platform

Also, called multi-platform is the way Java programmes can be used across different platforms e.g Apple and Window systems. Programmers using Java can develop software once than can run on a variety of systems.

According to Javaworld.com;[[2]](#footnote-2)





Keywords **-** these are *special words reserved* for constructing Java programmes which the developer cannot use when **naming** anything else.

Words such as;

**boolean, class, default, break, char, catch, double, final, if, private, public, float, long, short, switch, return, try and void – these and many more all have designated functions**.

|  |
| --- |
|  |

Variables– are the different data types used to identify *the use* of an object or entity.

There are four main ***primitive data*** types - integer, real number, boolean and character.

Numeric (integers) *whole numbers*

* int – this determines that only whole numbers will be used for this variable.

Minimum value of -2 (power 31) to 2 (power 31).

* byte – whole numbers - has minimum value of -128 and a maximum value 127.
* short – whole numbers - minimum value of -32,768 and maximum 32,767.
* long – whole numbers – min of – 2(power 63) and max 2(power 63).

Numeric (Real Number) - this determines that ***decimal numbers*** will be used for this variable.

* double – +/- 4.941 x 10 –324 to +/- 1.798 x 10 +308
* float – +/- 1.402 x 10 –45 to +/- 3.403 x 10 +38

### Boolean

* boolen – true / false (used for conditions where they can be either true or false)

### Textual

* String - group of text or name e.g. “John Hutton” – always inside quotation marks.
* char – these are letters both individual e.g. ‘b’ or ‘e’ – inside single quotes.

### Literal

*A literal is a fixed value that is given by the developer to any primitive variable.*

*e.g ;*

int acc = 21;

String colour = “red”

## Declaring Variables

Variable must be declared and initialized if the program is to run successfully.

Int a = 10, b = 12; //An example of initialization

Byte B = 22; //Initializes byte type for variable B.

Double pi = 3.1415 //Declares and assigns a value for PI.

Char a = ‘a’; //char variable is initialized with value ‘a’.

Identifiers – These are the names given to the specific data type for recognition by the developer. There is certain convention to naming these particularly ‘case sensitivity’.

Classes – are named usually with nouns with mixed case with the first letter of each word in capitals **e.g LibraryAccount**

Methods – should be verbs (doing words), these names are in camel case **e.g getPrice ();**

Constants – these are variables with assigned values that remain and cannot be changed, these names are written in capitals and separated by underscores,

e.g

final int MAX\_WIDTH = 10;

Or

final double PI = 3.142;

If for instance a different value is entered into the program

e.g PI = 22.5; // An error will be declared!

Operators – these perform operations such as;

* = (equals) && = And
* + - (plus and subtract) || = Or
* \* / (multiply and divide)
* % (modulus – finds remainder after division)

**The normal algebraic rules of precedence apply and brackets will be used for groups of expressions.**

* < > (greater than or lesser than)
* == (is equal to) a--; //short version of a - 1
* ! = (not equal to) a++ ; //short version of a + 1

### Example of Arithmetic Operations

int x, y, z;

x = 2;

Y = 3;

z = x + y; //**Answer is z = 5;**

## Input from the Keyboard

When input is required, the scanner class is used and is imported from **Java.util** (library)

**input.next();** // this allows input by an integer

**input.nextDouble** //this allows input by a double (real number)

## Commenting

Comments are included in the coding so that the developer or future programmers can remember what each piece of code refers to also so that Javadoc reports can be produced.

**Single line comments are comments that take only a single line;**

System.out.println("Customer Balance £ " +acc.getBalance();// get balance from object GasAccount

**Multiple line commenting are displayed like this;**

/\*\*

\* This is the main method that will contain the menu for the program. It will

\* offer a number of options for the user’s guidance. It will call on methods in

\* the object classes 'GasAccount & BusinessAccount' and will draw on inheritance a

\* & overriding methods.

\* @param args

\*/

## Indentation

Everything inside curly braces are indented by one TAB position. This make the code more readable and easier to discern.

public class Exam1 {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

final int PASSMARK = 40;

int mark;

System.out.print("Enter mark:");

mark = input.nextInt();

if (mark < PASSMARK) {

System.out.println("That is a fail.");

}

else {

System.out.println("That is a pass.");

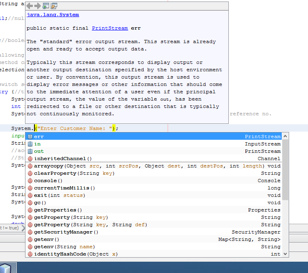
}

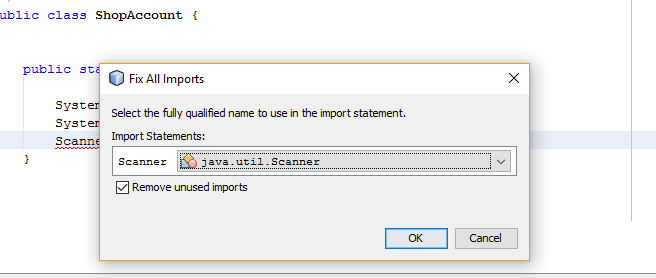
}

### Import Classes & Libraries

Java has prewritten files and classes that a programmer can use, by importing them from the libraries. This saves the programmer time in writing code.

Some of these are; Java. Lang , Java.util, Java. Util.concurrent These libraries are accessed with the dot operator, e.g of library is the System.out.print or acc**.**getDetails();

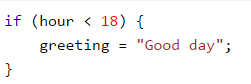




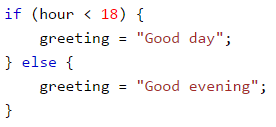
Selection in Java

Java has two means of deciding in a program. The ‘If’ statement and the ‘switch’ statement.

if - this statement is a conditional statement so that a certain action will be carried out if the condition is met. If hour is less than 18 then the String “Good day” will be output.



if else statement – is used where the condition is not met and you want another output.



### Nested if else Statements

Sometimes ‘if’ statements ‘if else’ statements are replaced with nested ‘if else’ statements.

A set of **if** statements may sometimes be replaced by a single construct which consists of *nested* **if else** statements. *This is more efficient and it means only the first piece of code needs to be executed.*

**Example**

Scanner input = new Scanner(System.in);

System.out.print("Enter a choice between 1 and 4:");

int menuChoice = input.nextInt();

if ( menuChoice == 1 )

{

System.out.println("You typed ONE"); ;

} // end of if statement

else if ( menuChoice == 2 )

{

System.out.println("You typed TWO" ) ;

} // end of if statement

else if ( menuChoice == 3 )

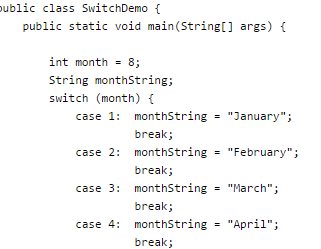
{

System.out.println("You typed THREE" ) ;

} // end of if statement

else if ( menuChoice == 4 )

switch - is where there are several options to be chosen from. If a certain value passing into the programme then that code is executed.



## Iteration in Java

Iteration allows for the repeating the same set of instructions, these are called ‘loops’. Three types - While loop, do while loop and the for loop.

Instead of;

System.out.print (“I love Java”);

System.out.print (“I love Java”);

System.out.print (“I love Java”);

## The While Loop

The while loops allow at least one execution of the condition first;

**class WhileLoop**

**public static void main(String args[** {

              int i= 1;

              while(i<=3)

                     {

                             System.out.println(i);

                              i++;

                       }

            }

    }

The output of the above code will be :-

1

2

3

## Do While Loop

**The ‘do while loop’ allows for condition to be checked after the first execution of the loop.**

Do

WhileLoop

   {

       public static void main(String args[])

          {

                     int i=1;

                     do

                      {

                              System.out.println(i);

                               i++;

                          }

 while(i<=3);

        }

}

The output of the above code will be :-

1

2

3

## The for Loop

So, the initial value, then the condition, then the increment will be executed

for (initial value; condition; increment) { then statement

}

**class**

For

Loop

   {

       public static void main(String args[])

          {

 for(int i=1;i<=3;i++)

                    {

                                   System.out.println(i);

                                }

                }

    }

The output of the above code will be :-

1

2

3

Arrays

An array is way of storing a **list** of values of the same data type, it could be strings, integers or real numbers. Different functions can be carried out with this list, for instance finding the average, the largest or smallest value etc.

e.g

**An array has an identifier in this case ‘count’**

int [] count = { 22 , 45 , 12 , 78 } ; This creates the array count as an integer array containing four integers and initializes them with values.

**This is an array count and with four integers and initializes them.**

count [ 0 ] holds 45 count [ 1 ] holds 56

count [ 2 ] holds 21 count [ 3 ] holds 99

count [ 0 ] holds 22 count [ 1 ] holds 45 count [ 2 ] holds 12 count [ 3 ] holds 78

# Object Orientated Java (OOP)

## Classes

There are three type of classes in Java;

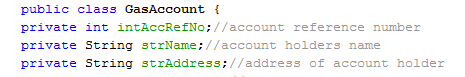
Programme class – this is the class that contains the main method for running the programme. It can also contain other methods which can be called into the main method.

Library class – This class is where all the methods are stored for use, they can be called by importing them from the library. All member of the library class use the **static** keyword.

The Object class – is the description of an object (a template) built to be used in conjunction with the programme class through the creation of an instance.

**Its main components are;**

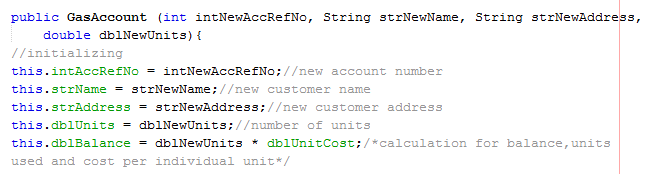
Properties– These are the attributes that are known about the object in this case for a bank account e.g;



**//private** means that these properties will not be accessible to other classes.

Methods & Types – ***Methods*** are the operations that will be carried out with the object.

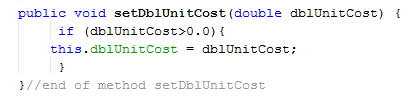
Constructor – this is the first method that is created and is a ***special method*** where all the properties that will be used are to be initialized and used within the parameters. There can be more than one constructor, e.g



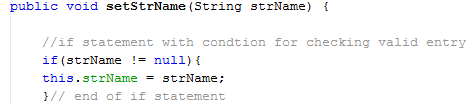
Getter Methods – these methods return data that is stored (they get this information and display it).



Setter Methods – these methods reset existing values.



Void Methods – these methods carry out calculations but don’t return any values.



# Object Class & Programme Class

After the object class properties and methods has been created they cannot be implemented without the programme class. This because the programme class contains the main method for executing the programme. Therefore, objects properties and methods must be called into the main method for the programme to work.

## Instance

This is achieved by creating an instance of an object as the object itself is only a template.

**Below an example of the creation of the instance of an object, actual values are now being inserted into the parameters, in our program.**

BusinessAccount testBus = new BusinessAccount (39583,"Saiorse Hamilton","99 Eagletop Way",7892.0,10.0);

**This was the constructor in the BusinessAccount the template for these values;**

public BusinessAccount (int intNewAccRefNo, String strNewName, String strNewAddress,

double dblNewUnits, double dblNewDiscount){

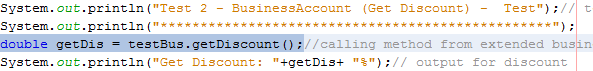
## Calling the method and dot operator

The instance is then used in conjuction with the method by the use of the dot operator.

So, the new instance , ‘testBus’ - is added to the method required which is getDiscount()

This is joined by the use of the dot operator e.g ‘testBus.getDiscount(); ‘

This will call the method getDiscount() from the object class and use this method for the program.



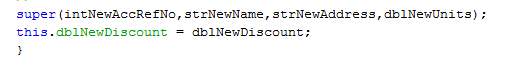
## Inheritance

With the creation of an object class with properties and behaviours, another **subclass** can **inherit** these properties and methods. This new subclass which is a new object can have new properties and new behaviours (methods). The original object is known as the **superclass.** This is a way of not having to reinvent the wheel by using the original object (template) and just adding a few more new characteristics (subclass).

The inheritance is implemented using the keyword, ‘**extends’.** Extends means that the new object is a subclass of the original object class or superclass.



The methods which we require to be extended to the new object from the superclasses constructor must be called into the new subclass, this is done using the keyword, **‘super’**, e.g

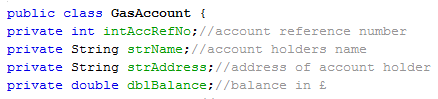


## Encapsulation

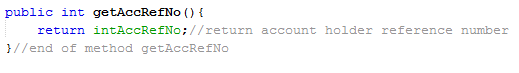
This is fundamental to how OOP works. Encapsulation is a way of **hiding the variables** and only allowing access to them through the methods.

This is done by declaring the variables as **private.** And declaring the methods as **public.** This gives access to these variables only through getter and setter method in the same class.

private keyword



public keyword



## Overloading/Overriding

Overloading allows a class to have two or more methods with the same name. The reason may be to have parameters list that are different for wanting similar or different information to be display or calculated in your program. These methods will of course have different functions. Overloading is also called static polymorphism.

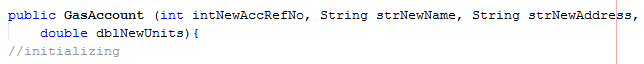
There are rules for this, they must have either;

* different parameters
* different data types
* data types must be in a different sequence.

Method GasAccount has three parameters.



Method GasAccount Has extra parameter - dblNewUnits



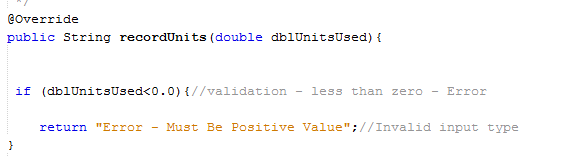
## Method Overriding

A method that has been created in the subclass that is also in the superclass is known as method overriding. It is used to provide a specific implementation of the method that is already in the superclass. It means that this method can be given precedence over the other as there may be a new element required in this method for the user/programmer.

Rule for this;

1. method must have same name as in the parent class
2. method must have same parameter as in the parent class.

The override annotation is used to confirm this;

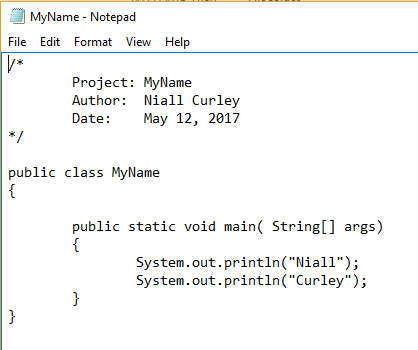


# Evaluate CLI with IDE

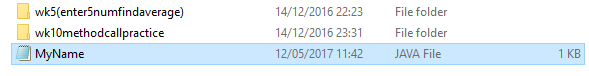
Computers are useless unless they receive instructions. The way a computer receives instructions are through a ‘user interface’. The two most commons way are through the Command Language Interface (CLI) and through the Integrated Development Environment (IDE).

## CLI

The Command Language Interface is used in conjunction with a simple code editor (Notepad). First the file is created and coding added. It is saved as a **.java file** and saved on the **C drive**. This is called the **Source Code.**



**Saved in C Drive**



## Compiler

The source code which contains all the commands, keywords and syntax are what something called ***the compiler*** uses to change it into bytecode.

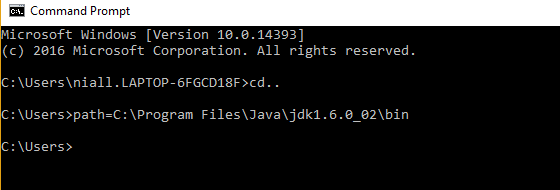
This is done by the java virtual machine (JVM), it is the interpreter that interprets the code that is written into the editor into machine code.

## Javac

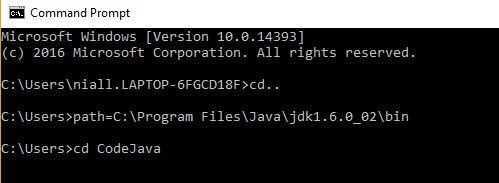
Javac is the command that is used to compile the code. Open the command prompt and type in the ***path command*** this will enable the search for the java compiler

**path=C:\Program Files\Java\jdk1.6.0\_02\bin**

will give you a clear prompt like this:



Type in, cd CodeJava



Then enter, java.MyName.java and press Enter. This will create a new file called MyName.class or an error will appear if there have been any type errors.

After the compilation of the program, the JVM is ready to **run** the program***.*** All methods and class line by line.

At the next prompt type in, java MyName and press Enter

This will display

Niall

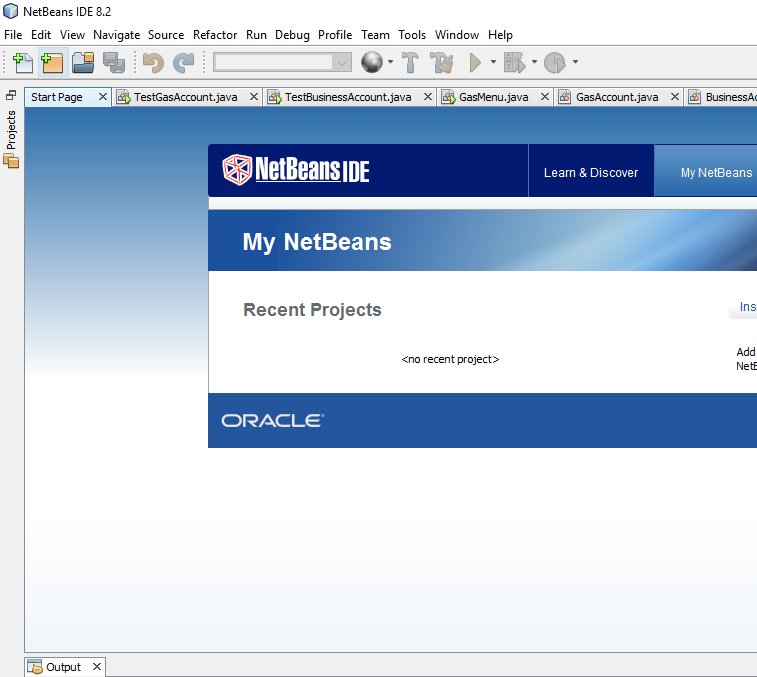
Curley

When you want to change your program in any way, you must go through this process of compiling, saving and executing in this way.

## Integrated Development Environment (IDE)

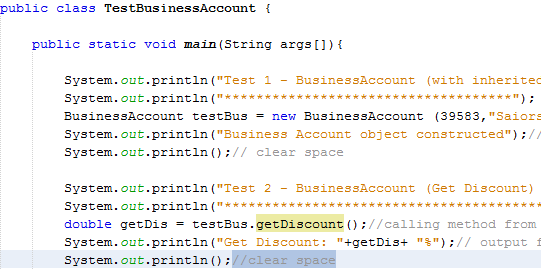
The use of an IDE which can be purchased online helps to facilitate the process of coding for the creation of Graphical User Interface (GUI). It is a toolbox that makes the writing of code easier.

Netbeans is one such IDE:



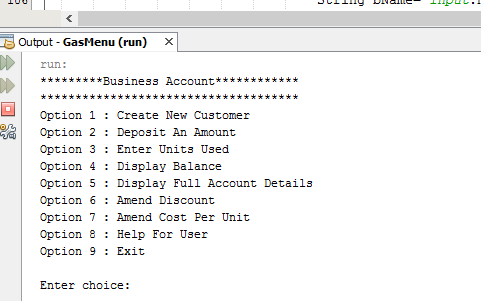
## IDE Colour Coding

As you can see the keywords are coloured in ‘blue’, the outputs with quotations are ‘red’ and the commenting is in ‘grey’



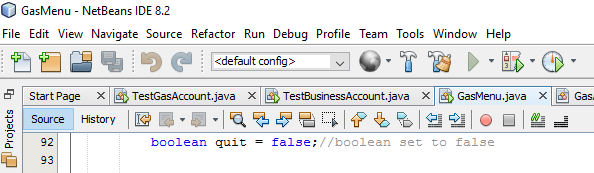
## Run & Output

Bottom Output section for displaying ‘Output – GasMenu (run)

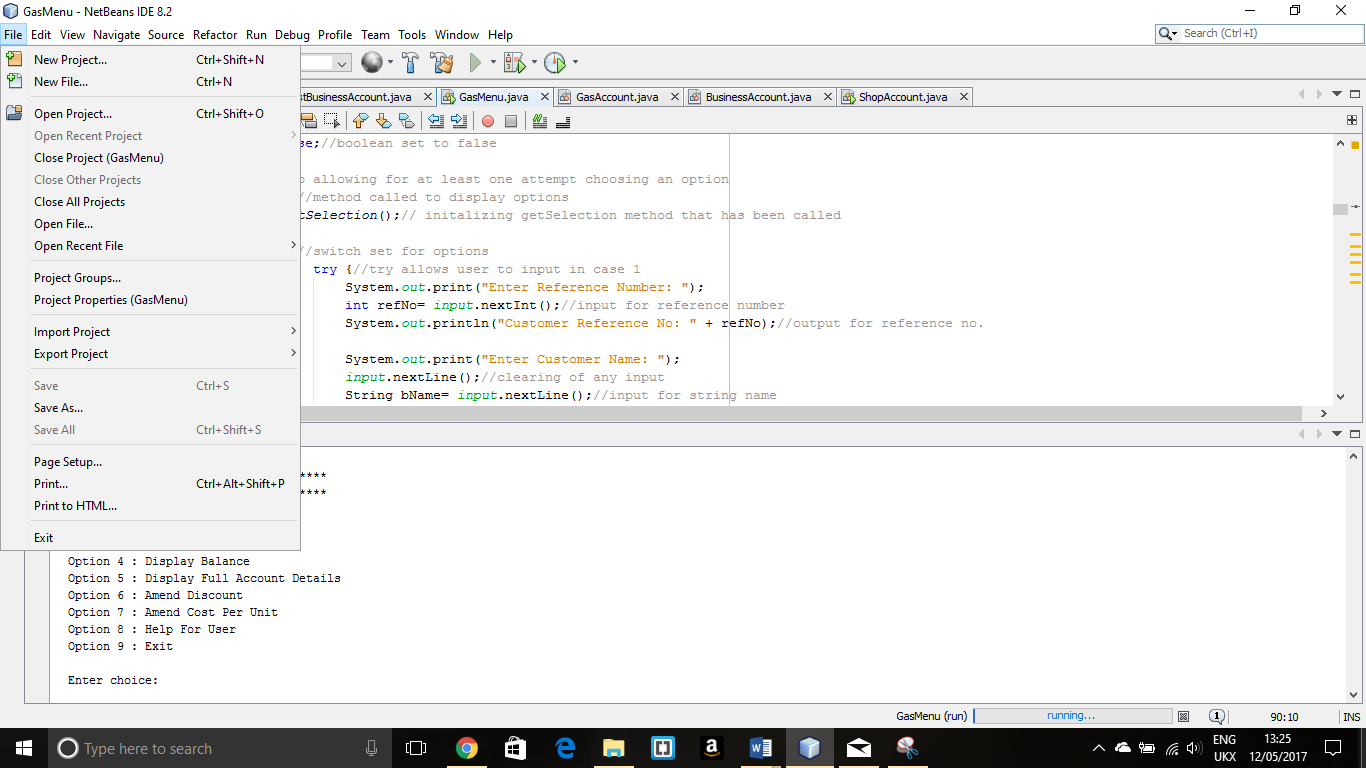


## Top Menu

This tool bar and drop-down menu gives the user multiple usages. Having menus in both written form and icon form.



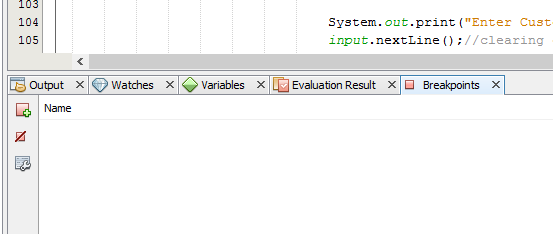
‘File’ facilitates the creation of new projects, new files, to save and print



The ability to **Edit** and paste files from past projects. **View** gives the developer the ability to customize the IDE to suit your needs (rearrange the toolbar).

#### Debug

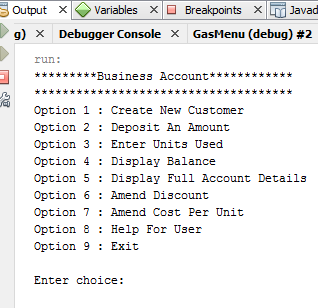
This tool facilitates fixing problems with the programme and runs a report assisting the developer.



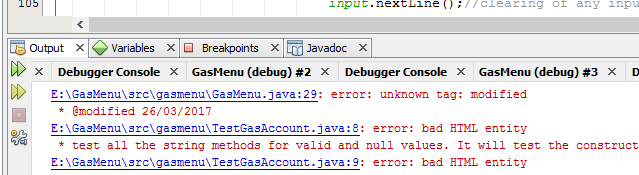
#### Run

This tool allows you to run your program and also has various sections including the Javadoc section. This helps in generating a report on the comments for each of the classes, methods and variables for the programmer.

Simple run programme:

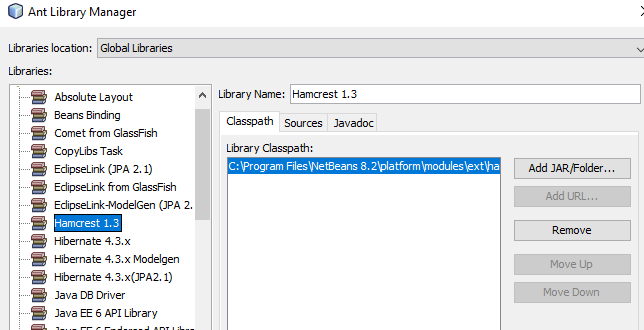


#### Javadoc report:



## Java Libraries

This is where all the files and prewritten java classes used for programming are kept and sourced.



## Main Differences Advantages & Disadvantages

In my opinion;

Using a just one program (IDE) is quicker than using two (code editor & command) as all the features are on one screen.

The colour coding of the different features in the IDE are better than the code editor/command (CLI) as it is easier to see when you have entered word that is only to be used as a keyword. You can spot errors easier with the red colour and fix it with the hints that are displayed on the screen. Rather than writing the code onto and editor and compiling it to find out this error.

In the IDE the menu and tools available for running the program are both in text and in icon form give you both options and also the ability to customize the screen to suit yourself. This is not available on the notepad.

With the IDE the developer can move around the different classes and change code when they need to. The code editor is much slower due to having to repeat the process when you need edit code or make changes.

With the IDE you can create multiple files and classes whilst working on different programmes, this is not the case with the editor/command method.

There is a Debugger built into the IDE which is not with the editor/command method.

You can run reports with the Javadoc to see the comments on the various methods & classes, you cant do this with the editor/command methods.

The NetBeans (IDE) library is useful for importing any methods and classes necessary for the developing of the program, this speeds up the coding process for the developer.

In saying this the editor/command method does teach the programmer to remember all the correct coding and this is lost on the IDE with all the help you get.

I think that the IDE is a better way to code as it speeds up the process and makes programming easier.

# Bibliography

Javaworld.com, 8.8.17, 10.56.

http://www.javaworld.com/article/2077249/client-side-java/client-side-java-java-for-embedded-systems.html

NetBeans, 12.5.17.  
https://netbeanside.secursoft.net/?network=ContentSearch&utm\_source=bing&utm\_medium=cpc&utm\_campaign=UK%3A%20Development%20Software&utm\_term=NetBeans%20IDE&utm\_content=netbeans-ide

Techmint, 6.5.2017, 11.36.

<https://www.tecmint.com/what-is-java-a-brief-history-about-java/>

1. https://www.tecmint.com/what-is-java-a-brief-history-about-java/ [↑](#footnote-ref-1)
2. <http://www.javaworld.com>/ [↑](#footnote-ref-2)