**Main Features of JAVA**

**Java is a platform independent language**

To understand the meaning of platform independent, we must need to understand the meaning of platform first. A platform is a pre-existing environment in which a program runs, obeying its constraints, and making use of its facilities.  
Lets back to the point. During compilation, the compiler converts java program to its byte code. This byte code can run on any platform such as Windows, Linux, Mac/OS etc. Which means a program that is compiled on windows can run on Linux and vice-versa. This is why java is known as platform independent language.

### Java is an Object Oriented language

Object oriented programming is a way of organizing programs as collection of objects, each of which represents an instance of a class.

4 main concepts of Object Oriented programming are:

1. [**Abstraction**](http://beginnersbook.com/2013/03/oops-in-java-encapsulation-inheritance-polymorphism-abstraction/)
2. [**Encapsulation**](http://beginnersbook.com/2013/05/encapsulation-in-java/)
3. [**Inheritance**](http://beginnersbook.com/2013/05/java-inheritance-types/)
4. [**Polymorphism**](http://beginnersbook.com/2013/03/polymorphism-in-java/)

### Simple

Java is considered as one of simple language because it does not have complex features like Operator overloading, Multiple inheritance, pointers and Explicit memory allocation.

### Robust Language

Two main problems that cause program failures are memory management mistakes and mishandled runtime errors. Java handles both of them efficiently.  
1) Memory management mistakes can be overcome by garbage collection.  Garbage collection is automatic de-allocation of objects which are no longer needed.  
2) Mishandled runtime errors are resolved by Exception Handling procedures.

### Secure

It provides a virtual firewall between the application and the computer.  Java codes are confined within Java Runtime Environment (JRE) thus it does not grant unauthorized access on the system resources.

### Java is distributed

Using java programming language we can create distributed applications. RMI(Remote Method Invocation) and EJB(Enterprise Java Beans) are used for creating distributed applications in java. In simple words: The java programs can be distributed on more than one systems that are connected to each other using internet connection. Objects on one JVM (java virtual machine) can execute procedures on a remote JVM.

### Multithreading

Java supports [**multithreading**](http://beginnersbook.com/2013/03/multithreading-in-java/). It enables a program to perform several tasks simultaneously.

### Portable

As discussed above, java code that is written on one machine can run on another machine. The platform independent byte code can be carried to any platform for execution that makes java code portable.

Example:

class WelcomeMessage

{

printMessage()

{

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

}

}

class Myclass

{

public static void main(String  []args)

{

WelcomeMessage obj=new  WelcomeMessage ();

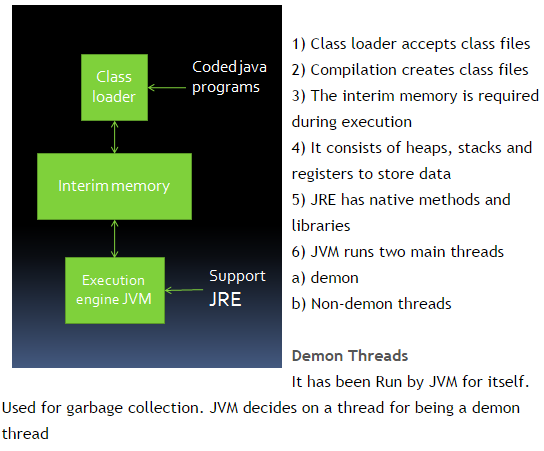
obj.printMessage();

}

}

**Output:** Hello World

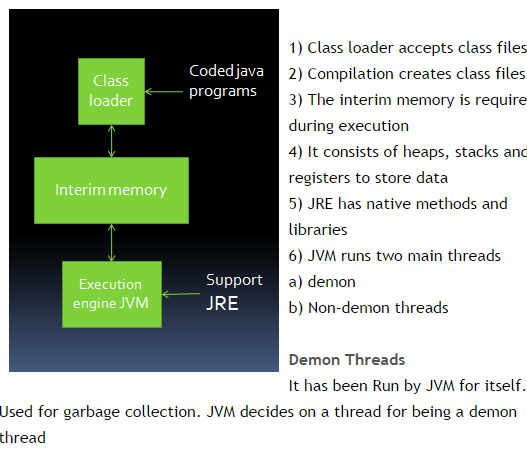
# Java Tutorial – JVM (Java virtual Machine)



#### Non-demon threads

main() is the initial and non-demon thread. Other implemented threads are also non-demon threads. The JVM is active till any non-demon thread is active.

# Java Tutorial – JVM (Java virtual Machine)



#### Non-demon threads

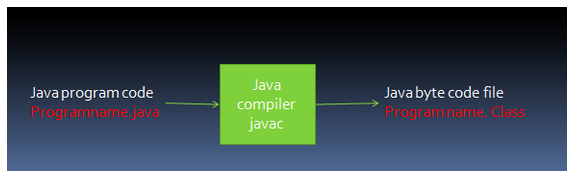
main() is the initial and non-demon thread. Other implemented threads are also non-demon threads. The JVM is active till any non-demon thread is active.

## Execution on JVM

1) JVM executes Java byte codes  
2) Other programming language codes if converted to adequate Java byte code can be executed on JVM  
3) JVM is different for different platforms and can also act as a platform itself  
4) JVM supports automatic  error handling by  intercepting the errors which can be controlled  
5) This feature is useful in platform independency and multi user ability of Java.

### Compilation

1) The compiler requires to know the TYPE of every CLASS used in the program source code  
2) This is done by setting a default user environment variable CLASSPATH  
3) The Javac (Java Compiler) reads the program and converts it into byte code files called as class files



#### Java Source code

1. It essentially consists of a main() method  
   2) This method is public and thus can be called by any object  
   3) This method is also static and so can be called without instantiating the object of the class  
   4) It does not return any value  
   5) The controlling class of every Java application usually contain a main method  
   6) This can be avoided to allow the class to be tested in a stand-alone mode.  
   7) Other methods can subsequently be called in main()

### Syntax of for loop:

for (initialization; condition; increment/decrement) {

statement(s) //block of statements

}

Mind the semicolon (;) after initialization and condition in the above syntax.

**Initialization expression** executes only once during the beginning of loop  
**Condition(Boolean Expression)** gets evaluated each time the loop iterates. Loop executes the block of statement repeatedly until this condition returns false.  
**Increment/Decrement** It executes after each iteration of loop

### For loop example to iterate an array:

Here we are iterating and displaying array elements using the for loop.

class ForLoopExample3 {

public static void main(String args[]){

int arr[4]={2,11,45,9};

//i starts with 0 as array index starts with 0 too

for(int i=0; i<4; i++){

System.out.println(arr[i]);

}

}

}

Output:

2

11

45

9

### Syntax of while loop

while (Boolean expression) {

statement(s) //block of statements

}

### While loop example

class WhileLoopExample {

public static void main(String args[]){

int i=10

while(i>1){

System.out.println(i);

i--;

}

}

}

### Infinite while loop

class WhileLoopExample2 {

public static void main(String args[]){

int i=10

while(i>1)

{

System.out.println(i);

i++;

}

}

}

Here is another example of infinite while loop:

while (true){

statement(s);

}

Here we are iterating and displaying array elements using while loop.

class WhileLoopExample3 {

public static void main(String args[]){

int arr[4]={2,11,45,9};

//i starts with 0 as array index starts with 0 too

int i=0;

while(i<4){

System.out.println(arr[i]);

i++;

}

}

}

**Output:**

2

11

45

9

### Introduction to do-while loop

do-while loop is similar to while loop, however there is a single difference between these two. Unlike while loop, do-while guarantees at-least one execution of block of statements. This happens because the do-while loop evaluates the boolean expression at the end of the loop’s body. Therefore the set of statements gets executed at-least once before the check of boolean expression.

**Syntax of while loop:**

do{

statement(s) //block of statements

}while (Boolean expression);

### do-while loop example

class DoWhileLoopExample {

public static void main(String args[]){

int i=10;

do{

System.out.println(i);

i--;

}while(i>1);

}

}

The output of this program is:  
10  
9  
8  
7  
6  
5  
4  
3  
2

### do-while loop example program to iterate an array

Here we are iterating and displaying array elements using do-while loop.

class DoWhileLoopExample2 {

public static void main(String args[]){

int arr[4]={2,11,45,9};

//i starts with 0 as array index starts with 0 too

int i=0;

do{

System.out.println(arr[i]);

i++;

}while(i<4);

}

}

Output:  
2  
11  
45  
9