**Table of Contents**

1. Introduction………………………………………………………………………….1
2. First Java Program………………………………………………………................4
3. FUNCTIONING OF TRAP EVENT COLLECTOR……………………………………..5

**Introduction to Java programming**

JAVA was developed by **Sun Microsystems** Inc in 1991, later acquired by **Oracle Corporation**. It was conceived by **James Gosling** and **Patrick Naughton**.

**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. Let’s 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, and Mac/OS etc. This 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:

* [**Abstraction**](http://beginnersbook.com/2013/03/oops-in-java-encapsulation-inheritance-polymorphism-abstraction/)
* [**Encapsulation**](http://beginnersbook.com/2013/05/encapsulation-in-java/)
* [**Inheritance**](http://beginnersbook.com/2013/05/java-inheritance-types/)
* [**Polymorphism**](http://beginnersbook.com/2013/03/polymorphism-in-java/)
* **Simple** (No Complex features like operator overloading, pointers, multiple inheritance, explicit memory allocation)
* **Robust Language** (Memory Management and Exception Handling)
* **Secure** (It provides a virtual firewall between application and computer)
* **Java is distributed**
* **Multi-Threading** (enables a program to perform several tasks simultaneously)
* **Portable.**

**First Java Program**

public class FirstJavaProgram {

public static void main(String[] args){

System.out.println("This is my first program in java");

}//End of main

}//End of FirstJavaProgram Class

* In the above program the class FirstJavaProgram has public access and hence declared public.
* ‘class’ is the keyword used to create a class.
* For running stand-alone programs ‘main’ method is needed which has a signature similar to the one defined in the above program.
* ‘Main’ method takes an array of strings as an argument. The name of the array can be anything.
* To display the output, pass the string as an argument to the method System.out.println.

**Steps for compilation and Execution**

**Step1:** Save the source file as WelcomeApp.java.  
**Step2:** Open command prompt and navigate to the directory where you have stored the file.  
**Step 3:** To compile, type javac FirstJavaProgram.java and press Enter.  
**Step 4:** On successful compilation, you will see the command prompt and FirstJavaProgram.class file in the same folder where WelcomeApp.java is stored. This is the byte code of the program.  
**Step 5:** To execute, type java FirstJavaProgram. Do not type the extension while executing.  
**Step 6:** See the output “This is my first program in java” displayed on the console.

### Common programming Errors in Java

The following are the general programming errors and the solution for them while running on windows machine.

1) **‘javac’ is not recognized as an internal or external command, operable program or batch file** – This means that the operating system cannot find the compiler – javac.  In order to resolve this the PATH variable has to be set.  
2) **Exception in thread “main”java.lang.NoClassDefFoundError: FirstJavaProgram –** This error means that java cannot find your compiled byte code, WelcomeApp.class. If the class file is present in directory other than the directory where java file is stored, then the CLASSPATH must be set pointing to the directory that contains compiled class files.

## Another Example

If you are a beginner and feel hard to understand the below example then just skip it and try to understand it once you finished reading all of my linked tutorials. After reading all tutorials it would be easy for you to learn things much faster.

1. package FirstCode
2. import java.lang.\*;
3. class WelcomeMessage
4. {
5. printMessage()
6. {
7. System.out.println("Hello World");
8. }
9. }
10. class Myclass
11. {
12. public static void main(String  []args)
13. {
14. WelcomeMessage obj=new  WelcomeMessage ();
15. obj.printMessage();
16. }
17. }

#### Interpreting the code

a) Line 1. The package FirstCode creates a folder to store the class files generated after compilation  
b) Line2. It imports the class library java.lang and its subsequent classes  
c) Line 3. Initiates a class with the name WelcomeMessage  
d) Line 5. Declares a method with name printMessage  
e) Line 7. Defines the actual working code of the method  
f) Line 10. Initiates the class having the main method; it should bear the name of the file : Myclass.java  
g) Line 12. Declares the main method  
h) Line 14. Initiates the creation of the object  
i) Line 15. Calls the method printMessage () with the help of the object  
j) The above code is saved and compiled to run on JVM

#### The programming structure

1) The programming pattern is divided into classes which has meth0d definitions  
2) This assists in distributing the code into smaller units  
3) The libraries can be used over and over again  
4) These codes generated here can be called in another program if required  
5) The memory allocation is done only after the execution of the new keyword  
6) It gets easier to collect memory that does not has any future use

**JVM (Java virtual Machine)**

**Execution**

**Engine JVM**

**Interim memory**

Support

JRE

Code Java programs

**Class**

**Loader**

1) Class loader accepts class files.  
2) Compilation creates class files.  
3) The interim memory is required during execution.  
4) It consists of heaps, stacks and registers to store data.  
5) JRE has native methods and libraries.  
6) JVM runs two main threads.  
 a) Demon threads  
 b) Non-demon threads

#### Demon Threads

It has been Run by JVM for itself. Used for garbage collection. JVM decides on a thread for being a demon thread

#### 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 byte code file

Programname.class

Java program code

Programname.java

**Java**

**Compiler**

**javac**

**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 contains 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 ().

**Loops**

A **loop** statement allows us to execute a statement or group of statements multiple times. Different types of loops are,



1. **For Loop**
2. **While Loop**
3. **do-while Loop**

**For Loop**

**What is for loop?**

It executes a block of statements repeatedly until the specified condition returns false.

**Syntax:**

for ( initialization; condition; increment/decrement) { statement(s) //block of statements }

**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:

class ForLoopExample {

public static void main(String args[]){

for(int i=10; i>1; i--){

System.out.println("The value of i is: "+i);

}

}

}

Output:

The value of i is: 10

The value of i is: 9

The value of i is: 8

The value of i is: 7

The value of i is: 6

The value of i is: 5

The value of i is: 4

The value of i is: 3

The value of i is: 2

Infinite Loop example:

class ForLoopExample2 {

public static void main(String args[]){

for(int i=1; i>=1; i++){

System.out.println("The value of i is: "+i);

}

}

}

Example to iterate an array:

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

**While Loop**

**What is while loop?**  
Like for loop, it also executes a block of statements repeatedly until the condition (Boolean expression) returns false.

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 Loop example:

class WhileLoopExample2 {

public static void main(String args[]){

int i=10

while(i>1)

{

System.out.println(i);

i++;

}

}

}

Example to iterate an array:

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

**do-while Loop**

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.

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);

}

}

Example to iterate an array:

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