Steps to Design a First Application: -

Step-1: - ***Select an Editor.***
Step-2: - ***Write the application.***
Step-3: - ***Save the application.***
Step-4: - ***Compilation Process.***
Step-5: - ***Execution process.***

Step1: - Select an Editor

Editor is a tool or software it will provide very good environment to develop java application.
Ex: - Notepad, Notepad++, edit Plus…. etc.

IDE: - (Integrated development Environment)

IDE is providing very good environment to develop the application.
Ex: - Eclipse, MyEclipse, NetBeans, JDeveloper…. etc.

IDE is a real-time standard but don't use IDE to develop core java applications because 80% work is done by IDE & remaining 20 % work is down by developer.

80% work of IDE is: -

- ✓ Automatic compilation.
- ✓ Automatic package import.
- ✓ It shows all the predefined methods of classes.
- ✓ Automatically generate try catch blocks and throws (Exception handling)
- ✓ It is showing the information about how to fix the bug.....etc.

Note:- Do the practical's of core java only by using Edit Plus or Notepad++ software.

Step 2:- Write a program.

- Write the java program based on the java API (Application Programming Interface) rule and regulations.
  Open editplus --->file ---->new ----->click on java (it displays sample java application)
- Java is a case Sensitive Language so while writing the program you must take care about the case (Alphabet symbols).

Example application:-

```
import java.lang.System;
import java.lang.String;
class Test          //class declaration
{           //class starts
        public static void main(String[]args)       //program execution starting point
        {       //main starts
                System.out.println("hi Raman"); //printing statement
        }       //main ends
};          //class ends
class A
{
};
class B
{
};
```

*In above example **String & System c**lasses are present predefined java.lang package hence must import that package by using import statement.*

To import the classes into our application we are having two approaches,

1) Import all class of particular package.
    a. *Importjava.lang.*; //it is importing all classes of java.langpackage.*
2) Import required classes
    a. *Import java.lang.System;*
    b. *Import java.lang.String;*

In above two approaches second approach is best approach because we are importing application required classes.

Note: The source file is allows declaring multiple java classes.

Step3:- save the application.

- *After writing the application must save the application by using **(.java)** extension.*
- *While saving the application must follow two rules*
    - *If the source file contains public class then public class name & Source filename must be same **(publicClassName.java)**. Otherwise compiler generate error message.*
    - *if the source file does not contain public class then save the source file with anyname **(anyName.java)** like A.java , Raman.java, Anu.java …….etc. .*

Note: - The source file allowed only one public class, if we are trying to declare multiple public classes then compiler generate error message.

example 1: - invalid

```
//Raman.java
public class Test
{   };
class A
{ };
```

Application location:

-

```
D:                          (any disk)
|-->raman                   (any folder)
      |-->code.java         (your filename)
```

example 2: - valid

```
//Test.java
public class Test
{ };
class A
{   };
```

example 3: - invalid

```
//Test.java
public class Test
{ };
public class A
{ };
```

Step-4: - Compilation process.

*Compile the java application by using **javac** command.*

Syntax: -

Javac filename.java

Javac Test.java

Process of moving application saving location: -

C:\Users\hp> *initial cursor location*

C:\Users\hp>d: *move to local disk D*

D:\>cd raman *changing directory to raman*

D:\raman>javac code.java *compilation process*

Whenever we are performing compilation the compiler will check the syntax errors.

- *If the application contains syntax errors then compiler will generate error message in the form of compilation error.*
- *If the application does not contains syntax errors then compiler will generate .class files.(conversion of **.java** to **.class**)*

Note: - in java .class files generated by compiler at compilation time and .class file generation based on number of classes present in source file.

If the source file contains 100 classes after compilation compiler generates 100 .class files

The compiler generate .class file and .class file contains byte code instructions it is intermediate code.

Step-5:- Execution process.

*Run /execute the java application by using **java** command.*

Syntax:-

Java Class_Name

Java Test

Whenever you are executing particular class file then JVM perform fallowing actions.

- *JVM wills loads corresponding .class file byte code into memory.*
- *After loading .class file JVM calls main method to start the execution process.*

In above two cases if the class file or main method is not available then at runtime JVM will generate error message.

*If the main method is not available: **"Main method not found in class A, please define the main method".***

*If the .class is not available : **"Could not find main class**".*

*Executing all generated **.class** files based on example given in second step:-*

Test class	***--->***	*class is loaded & main is present*
A class	***--->***	*class is loaded but main is not present*
B class	***--->***	*class is loaded but main is not present*
XXX class	*--->*	*XXX class is not present*

D:\raman>java Test

Hi Raman

D:\raman>java A

Error: Main method not found in class A, please define the main method as: public static void main (String[] args)

D:\raman>java B

Error: Main method not found in class B, please define the main method as: public static void main (String[] args)

D:\raman>java XXX

Error: Could not find or load main class XXX

Note 1: - compiler is translator it is translating ***.java*** *file to* ***.class*** *where as JVM is also a translator it is translating* ***.class*** *file to* ***machine code.***

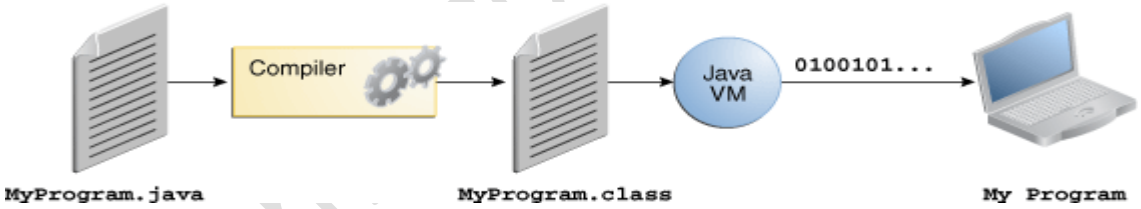
Note 2:-Compiler understandable file format is ***.java*** *file but JVM understandable file format is* ***.class*** *file.*

Note 3:- it is possible to compile multiple files at a time but it is possible to execute only one .class file at a time.

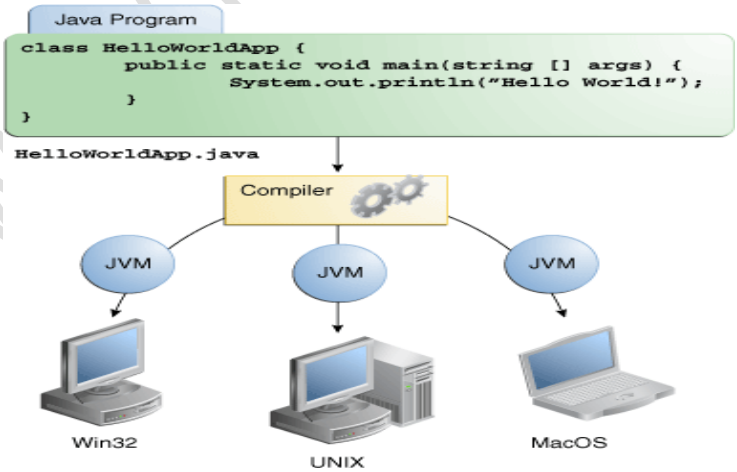
Note 4:- The .java file contains high level language (English) but .class file contains byte code instructions.

Note 5:- The source is allowed to declaring multiple non-public classes but only one public class.

Environment of the java programming development: -



First program development: -



Note: - java is a platform independent language but JVM is platform dependent.

Example-1:-

- *Java contains 14 predefined packages but the default package in java is **java. lang** package it means if we are importing or not by default this package is imported.*
- *In below example importing*

classes are optional.

```
class Test
{       public static void main (String [] args)
        {       System.out.println("hi raman");
        }
}
```

Example-2: -

*The class contains main method is called **Main class** and java allows to declare multiple main class in a single source file.*

```
class Test1
{       public static void main (String [] args)
        {       System.out.println("Test1World!");
        }
}
class Test2
{       public static void main (String [] args)
        {       System.out.println("Test2World!");
        }
}
class Test3
{       public static void main (String [] args)
        {       System.out.println("Test3World!");
        }
}
```

D:\raman>java Test1
Test1 World!

D:\raman>java Test2
Test2 World!

D:\raman>java Test3
Test3 World!