ABESIT
COLLEGE OF ENGINEERING

CRC-Training

Core Java

First Java Program

Step -1 Example

```
class A
{
    public static void main(String args[])
    {
        System.out.println("Hello World!");
    }
}
```

Step- 2 Save Code with class Name

A.java

Step-3 Compile the code

D:\> javac A.java (when you compile the code it will generate an intermediate code (Byte code) it is not a machine code.

Step-4 run the Program

D:\> java A (running the program by starting the JVM with the A.class. The JVM translates the bytecode into something the underlying platform understands , and runs your program.

ABESIT COLLEGE OF ENGINEERING

CRC-Training

Trace a Program Execution

Enter main method

```
//This program prints Hello World!
class A {
  public static void main(String[] args) {
    System.out.println("Hello World!");
  }
}
```

ABESIT COLLEGE OF ENGINEERING

CRC-Training

Trace a Program Execution

Execute statement

```
//This program prints Hello World!
class A {
  public static void main(String[] args) {
    System.out.println("Hello World!");
  }
}
```

Trace a Program Execution

```
//This program prints Hello World!
class A {
  public static void main(String[] args) {
    System.out.println("Hello World!");
  }
}
```

Command Prompt

```
E:\ABESIT>javac A.java

E:\ABESIT>java A
Hello World!

E:\ABESIT>
```

print a message to the console

Second Java Program

```java
// Program in Java to print welcome with your name.

class Main
{
    public static void main(String[] args) {
        String Name = "Ramesh";
        System.out.println("Welcome Mr. " + Name);
    }
}

```

```
Welcome Mr. Ramesh
```

Third Java Program

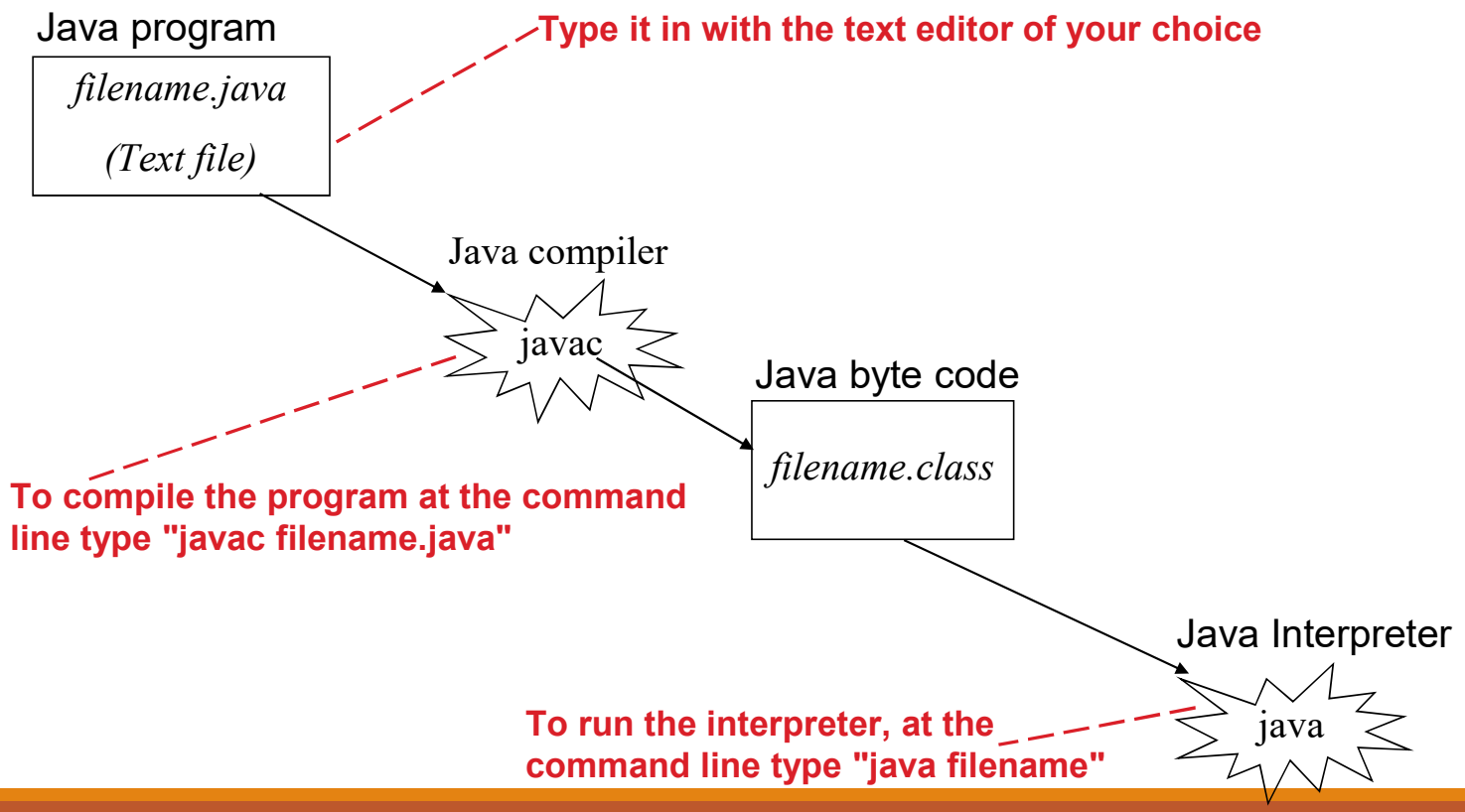
```java
// Program in Java to print sum of 2 integer number.

class Main
{
    public static void main(String[] args) {
        int num1 = 10;
        int num2 = 20;
        int sum = num1 + num2;
        System.out.println("sum is " + sum);
    }
}

```

```
sum is 30
```

Creating, Compiling And Running Java Programs



Compiling A Java Program

A.java

```
class A
{
    public static void main (String[] args)
    {
        System.out.println("Hello World!");
    }
}
```

Type "javac A.java"

javac

A.class

(Java byte code)

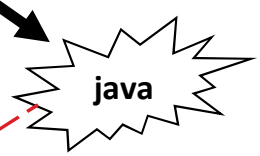
```
10000100000001000
00100100000001001
    :         :
```

Running A Java Program

A.class

(Java byte code)

```
10000100000001000
00100100000001001
    :         :
```



Type “java A”

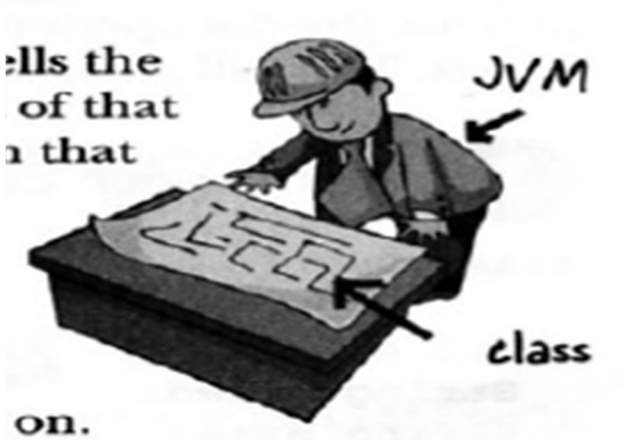
(Platform/Operating specific binary

```
10100111000001000
00100111001111001
    :         :
```

Program Observation

- **When the JVM starts running , it looks for the class that is at the command line. Then it starts looking for a specially written method that looks exactly like:**

```
public static void main(String args[])
{
    //Your Code Here
}
```



- **Note:** Every Java application has to have at least one class, and at least one main method.

Program Observation

You can define main method in the following ways

```
a) public static void main(String args[])
b) static public void main(String args[])
c) public static void main(String ...a)   // Allowed in JDK 1.5 or higher
d) static public void main(String ...a)  //Allowed in jdk1.5 or higher
```

- The **main** method is **public**, because java is **package centric language**.
- Now JVM has to access your main method from **outside your package**. That's why it set to as **public**.
- The **main** method also **static**, because static is loaded on the compile Time and **not required** to **instantiate**.

Program Observation

You can define main method in the following ways

a) public static void main(String args[])
b) static public void main(String args[])
c) public static void main(String …a) // Allowed in JDK 1.5 or higher
d) static public void main(String …a) //Allowed in jdk1.5 or higher

- The **main** method is **void** , because it is **not** required to **return any thing** to the **JVM**.
- **main** is the name of the method
- **String args[]** is used for **command line arguments**, where **String** is a **Java class**.

Program Observation

```
System.out.println("Hello Java ! ");
```

- where **System** is a **final** class, found in **java.lang** package.
- **Out** is a **object of PrintStream** class and it declare **static** in the **System** Class.
- **Println(String msg)** this is a method define in the **PrintStream** class. Used to print **on console** and print in new line.

Can we have more then 1 class in java program?

- Yes
- Ok, let see how program will work if we have more then 1 class.

Can we have more then 1 class in java program?

```java
// Program in Java for 2 classes.
class Main
{
    public static void main(String[] args) {
        String Name = "Shyam";
        System.out.println("Hello Mr. " + Name);
         A obj=new A();
         obj.Age(19);
    }
}
class A
{
    void Age(int age) {
        System.out.println("You are " + age + " year old");
    }
}
```

```
Hello Mr. Shyam
You are 19 year old
```

Can 1 java file have more then 1 public class?

- No

Why only 1 public class in Java file?

- So that the compiler is aware of the starting point.
- It forces all Java code to be *organized in a* ***certain way***, which in the long run helps improve *code* ***readability*****.**
- It make the compilation process **faster** because it enables a more efficient lookup of source and compiled files during linking.
- The Java designers chose a **strict approach** that enforces their idea of **good design** practices.
- The **name of the Java file** should be the same as the **name of the public class** in it.

ABESIT COLLEGE OF ENGINEERING

CRC-Training

What is 2 level of compilation in Java?

- Level 1 – Compiles the Source Code to Byte Code
- Level 2 – Interprets the Byte Code to Native Code

- **What is JIT?**
- A JIT compiler **compiles bytecode to native code .**

Java Coding Convention

- **Class and Interfaces** – The First letter should be capital, and first letter of the inner words should be Capital.

  Example : Account

  PrintWriter

  Emp

- **Methods** – The First Letter should be lowercase, and then normal camelCase rules should be used.

  Example : getBalance

  setCustomerName

- **Variables** – Same rule as Method Rule.
- **Constants** – All in Uppercase.

  Example : MIN_HEIGHT

ABESIT COLLEGE OF ENGINEERING

CRC-Training

Java Class and Object

What is Class in Java?

- **Class** is a blueprint or a set of instructions to build a specific type of **object**.
- It is a basic concept of Object-Oriented Programming which revolve around the **real-life entities**.
- **Class** in Java determines **how** an **object** will **behave** and **what** the **object** will **contain**.
- **Class** does **not** occupy memory.
- **Class** is a group of variables of different data types and a group of methods.

- **Class** contains

```
class Test {
        1) variables
        2) methods
        3) constructors
        4) instance block
        5) static block
}
```

Example:

- ✓ **Animal**
- ✓ **Student**
- ✓ **Bird**
- ✓ **Vehicle**
- ✓ **Company**

ABESIT COLLEGE OF ENGINEERING

CRC-Training

Java Class and Object

What is Object in Java?

- **Object** is an instance of a class.
- When JVM encounters the **new keyword**, it will use the same **class** to make an **object**.
- That **Object** will have its own states, and access to all of the behaviour defined by it's **class**.
- An **object** in OOPS is nothing but a **self-contained component** which consists of methods and properties to make a particular type of data useful.
- **For example** color name, table, bag, barking.
- When you send a message to an **object**, you are asking the object to invoke or execute one of its methods as defined in the class.
- **Object** has a **memory** location allocated.

Example:

- ✓ **dog**

Structure of Java Program

- **Class Components**
  - A class Consists of data members and methods

Class Name

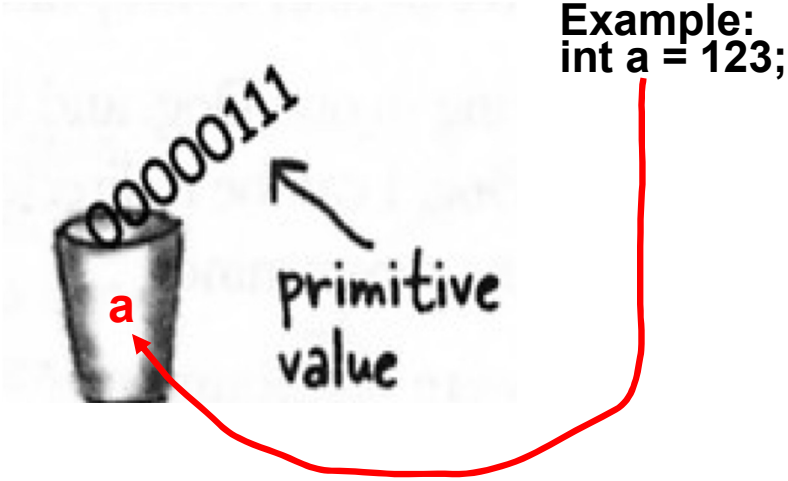
Variable declaration

Method1
Method2
.
.
Method n

Variables

- A variable is just like a cup or container. It can hold something. It has a size and a type.

Sample Example



ABESIT COLLEGE OF ENGINEERING

CRC-Training

Rules for Declaring a Legal Variable:

- Variable must start with a letter, $, _. Cannot start with number.
- After the first letter, contain numeric.
- No Limit to the number of characters an variable can contain.
- Cannot use keyword in variable name.
- Variables are case-sensitive.

For Employee Project – Employee_Id, Employee_Name, Employee_Salary variables

For Student Project – Student_Id, Student_Name, Student_Marks variables

Types of Variable:

- There are three types of variables in Java: **Local, Instance, and Static.**

Local variable:

- This is a variable that is declared inside the body of a method.
- This variable can be visible only within that method.
- A local variable cannot be defined with "static" keyword.

Types of Variable:

Instance variable:

- This Java variable is defined without the STATIC keyword.
- Declared inside the class but outside of a method declaration.
- They are object-specific (instance-specific) variables, which is why they are known by this name.

Types of Variable:

Static variable:

- This variable is declared as static.
- It is the variable that should be initialized first, especially before an instance variable is initialized.
- Static variables are shared.
- All instances of the same class share a single copy of the static variables.

Note : Instance variables 1 per instance.
Static variables 1 per class.

Methods:

- A method in Java is a block of code that, when called, performs specific actions mentioned in it.

How to Declare Methods in Java?

- You can only create a method within a class.

Syntax:

```
public int addNumbers (int a, int b){
 //method body
 }
```

Type of Methods:

Methods in Java can also be classified into the following 2 types:

- Static Method
- Instance Method

Static Method

- Static methods are the ones that belong to a class and not an instance of a class.
- There is no need to create an object to call it, and that's the most significant advantage of static methods.
- It is possible to create a static method by using the "static" keyword.
- The primary method where the execution of the Java program begins is also static.

Type of Methods:

Instance Method

- The instance method is a non-static method that belongs to the class and its instance.
- Creating an object is necessary to call the instance method.

```java
class A{
    static void display(){  // static method
        System.out.println("static method");
    }
    void show(){            // instance method
        System.out.println("instance method");
    }
    public static void main(String[] args) {
    A obj=new A();
    display();  // static method call
    obj.show();  // instance method call
    }
}
```

```
static method
instance method
```

Access of variable and method

```java
class A{
    int a;                      // instance variable
    static int b;               // instance variable
    static void display(){  // static method
        System.out.println("static method called");
    }
    void sum(){             // instance method
        int c = a + b;        // local variable
        System.out.println("in instance method c = " + c);
    }
    public static void main(String[] args) {
    A obj=new A();
    display();  // static method call
    obj.sum();  // instance method call
    }
}
```

```
static method called
in instance method c = 0
```