WELCOME TO

I- RISE SOFTWARE TRAINING INSTITUTE

‘Ambition is the path to success

Persistence is the vehicle you arrive in’

Syllabus:

1St Chapter:

Introduction

Identifiers

Rules to define Identifiers

Classes

Packages

Object

Data types

Constructor

Variables

Arrays

Access specifiers

Operators

Flow Controls:

If else

Switch

While loop

Do-while

For loop

For Each

2nd Chapter:

OOP's concept

Abstract Classes

Interfaces

Exception handling

String handling

Collection

Core Java:

History:

JAVA was developed by James Gosling at Sun Microsystems Inc. in the year 1991, later acquired by Oracle Corporation. It is a simple programming language. Java makes writing, compiling, and debugging programming easy. It helps to create reusable code and modular programs.

Sun Microsystems released its first public implementation in 1996 as Java 1.0. It provides no-cost -run-times on popular platforms.

Java is the name of an island in Indonesia where the first coffee(named java coffee) was produced.

Software Programming Language:

Software programming languages allow us to give instructions to a computer in a language the computer understands. Just as many human-based languages exist, there are many computer programming languages that programmers can use to communicate with a computer.

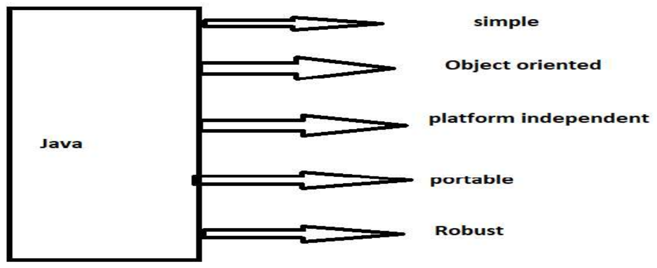
It is a set of instructions written in any particular language (C, C++, Java, and Python) to implement a definite task.

The portion of the language that a computer can understand is called a binary (byte code). Translating programming language into binary is known as compiling.

There are many software programming language like java, python, dot net, pearl, scala etc.

Java features:

Java has many important features that makes java one of the most useful programming language today. As the time passes, java has emerged as one of the most preferable programming language for different types of need. This tutorial will cover some of it's features which makes it one of the most useful language.



Simple:

Java is a simple language because it's syntaxes are very similar to C++ syntaxes, it also provides automatic memory management through garbage collection, programmers don't have to focus on complex memory management.

Also there is no concept of pointers in java that confuses programmers a lot. All these things makes java a simple language.

Object Oriented

Java is an object oriented programming language because java supports the principles of Object Oriented Programming(OOPs). Everything in java plays around objects, we create variables, methods for objects in a program.

Portable

Java is portable because the bytecode(.class file) of a program can be run on different machines without any change in it. Java was designed with a goal that if new architectures are developed, the java environment could be ported easily. While porting all you need to have is the .class files of application on existing machine. Just copy these .class file on new machine and run your application by installing(if not already installed) the java virtual machine on new machine.

Secured

As java does not have concept of pointer and provides the automatic memory management, it reduces the chances of memory leak. Also java program runs in a separate java virtual machine which enhances it's security. These features together make java one of the most secured language.

Robust

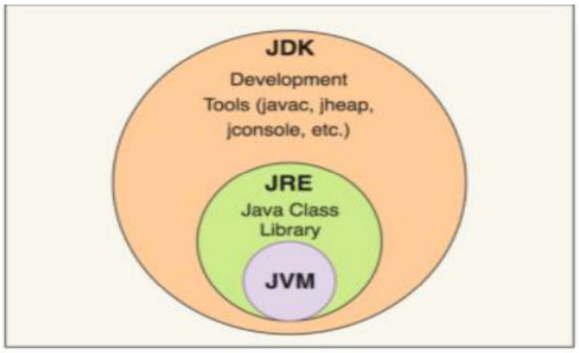
A language is called robust if it is reliable. The reliability comes because java strongly checks for error in program at compile and runtime both, to eliminate error-prone situations. The reliability also comes because java also has the strong memory allocation and automatic garbage collection mechanism which reduces the probability of crashing a program at runtime.

Platform Independent:

Java is platform Independent programming language because java program compiled code can run in all operating systems.

Overview of java development kit(JDK):

While we were using the term JDK, when we learn about bytecode and JVM . So, as the name suggests, it is a complete Java development kit that includes everything including compiler, Java Runtime Environment (JRE), java debuggers, java docs, etc. For the program to execute in java, we need to install JDK on our computer in order to create, compile and run the java program.



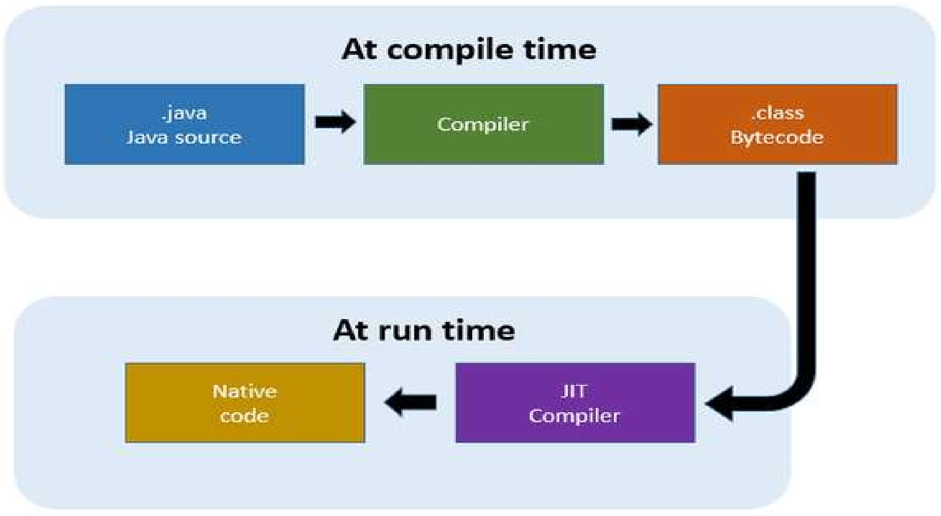
Java Runtime Environment (JRE):

JDK includes JRE. JRE installation on our computers allows the java program to run, however, we cannot compile it. JRE includes a browser, JVM, applet supports, and plugins. For running the java program, a computer needs JRE.

Compiler:

Compiler (javac):

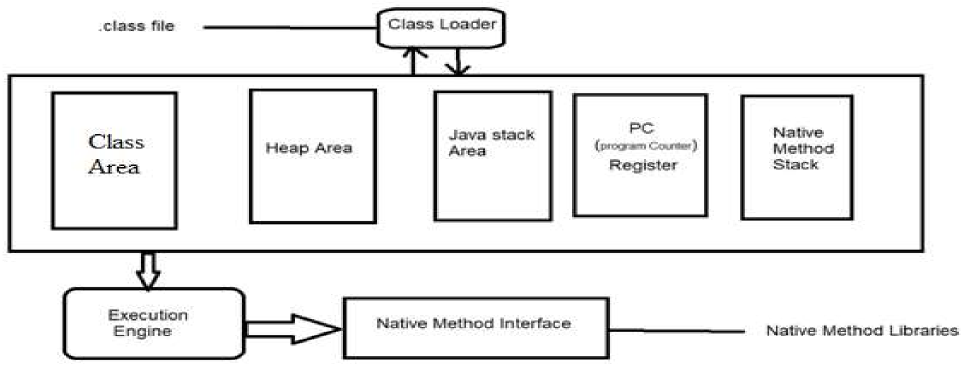
We are going to learn about the Java Compiler which is used to compile the Java program into byte code. The Java program (.java) file is checked for syntax error and then compiled into class file (.class) which can be interpreted and converted into the machine language in runtime. The Java interpreter (JVM) is used to convert .class file into machine language on the host machine where it is executed.



<u>Java Virtual Machine(JVM):</u>

Java Virtual Machine (JVM) is a platform it is responsible to run java bycode produced by the compiler by translating it into current operating system machine language. Every Operating System has a different JVM but the output they produce after the execution of bytecode is the same across all the operating systems (windows, Mac, Android).

Its provides runtime environment with an equally partitioned identical memory areas for executing java bycode in any OS.



1. **ClassLoader:-**

ClassLoader is responsible to load and store the given class bytecode into JVM from both the program and the Java API and also it is responsible to link the loaded class to the runtime application of JVM and initialize the loaded class memory.

There are three types of class loader:

1) Bootstrap class loader:-

- This ClassLoader is Responsible to load classes from jdk\jre\lib folder.
- All core java API classes present in rt.jar which is present in this location only. Hence all API classes (like String, StringBuffer, etc) will be loaded by Bootstrap class Loader only.
- That is Bootstrap ClassLoader is Responsible for Load Classes from Bootstrap ClassPath.
- Bootstrap ClassLoader is by Default Available with the JVM.
-

2) Extension ClassLoader:-

- It is the child of Bootstrap class loader which loads the classes from jdk\jre\lib\ext folder.
- This class loader is responsible to load classes from the extension classpath.

3) Application ClassLoader:-

- It is the Child of Extension ClassLoader.
- This ClassLoader is Responsible to Load Classes from Application Class Path (Current Working Directory).
- It Internally Uses Environment Variable Class Path.

2. **Execution Engine:-**
   - It is a type of software used to test hardware, software, or complete systems. The test execution engine never carries any information about the tested product.
   - It is a central component of JVM that communicates with various memory areas of JVM.
   - Execution engine executes bytecode which is assigned to the run data areas in JVM via classloader.
   - Java class file executed by the execution engine.
3. **Native Methods Interface (JNI) :**
   - The Native Method Interface is a programming framework. It allows Java code which is running in a JVM to call by libraries and native applications.
   - Native Method Libraries are libraries that are written in other programming languages, such as C, C++, and assembly. These libraries can be loaded through JNI.
4. **Native Method Libraries:**
   - Native Library is a collection of the Native Libraries(C, C++) which are needed by the Execution Engine.
5. **Class/ method Area:**
   - Classes are Loaded and stored in the Method area.
   - Static variable memory is created in the method area in their class context.
6. **Heap Area:**
   - The loaded class object is created in the Heap area.
   - Also, Non-static variable memory is created in the Heap area.
7. **Java Stack Area:**
   - Method logic executes in the java stack area inside the main method.
   - A separate runtime stack will be created for every thread.
   - All local variable memory is created in the Java stack area.
   - Its created multiple stack frames for every thread.
8. **PC Register:**

- Each thread will have separate PC Registers to hold the address of the current executing instruction once the instruction is executed the PC register will be updated with the next instruction.
- It is local to each thread and contains the address of the JVM instruction that the thread is currently executing.

9. **Native method Stack:**
    - Native Method Stack holds native method information. For every thread, a separate native method stack will be created.
    - Native method stacks hold the instruction of native code depending on the native library. It is written in another language instead of Java.

What is Object?

- An object is a real-world entity, an entity that has state and behavior e.g., chair, bike, marker, pen, table, car, etc.
- Object is an instance of a class.
- **State**:- Represents the data (value) of an object.
- **Behavior**: - Represents the behavior (functionality) of an object.

- For Example, Pen is an object. Its name is Reynolds; the color is white, known as its state. It is used to write, so writing is its behavior.

What is Instance?

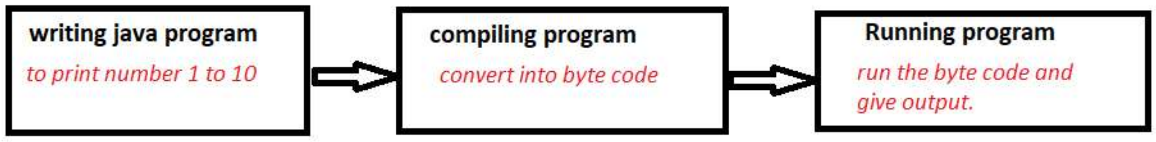
- An instance is a single, unique memory allocation of a class that represents an object
- Physically with a specific value.

What is a class in Java?

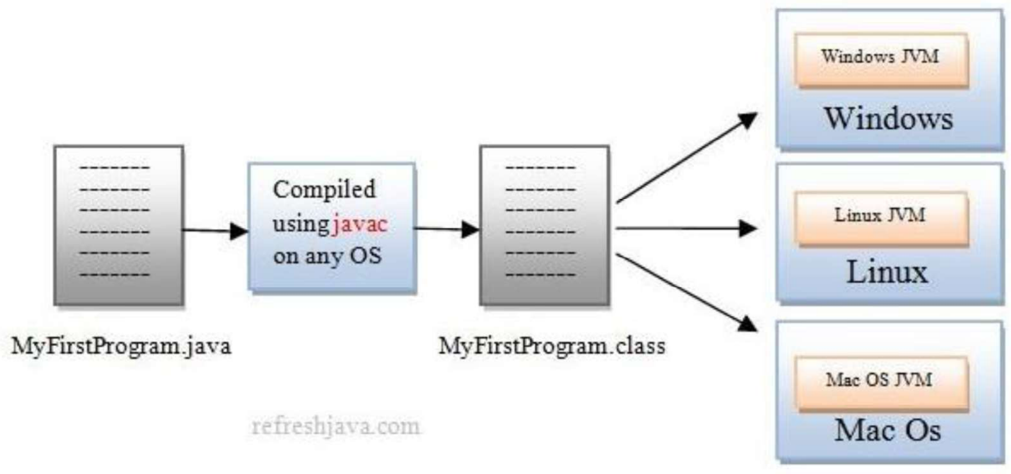
- It is a template or blueprint(design) from which objects are created. It is a logical entity. It can't be physical. It decides how the object should build.
- Class is a logical structure as an object has a physical reality.
- A class defines the structure, state and behavior that will be shared by a set of objects. Hence each object of a given class contains the structure, state and behavior defined by the class.

Program execution flow:

This is generally referred to as JVM. There are three execution phases of a program. They are written, compile and run the program.



- Writing a program is done by a java programmer like you and me.
- The compilation is done by the JAVAC compiler which is a primary Java compiler included in the Java development kit (JDK). It takes Java program as input and generates bytecode as output.
- In the Running phase of a program, JVM executes the bytecode generated by the compiler and give output.



<u>Identifiers</u>:

- A name in java program is called identifier.

- Identifiers means symbolic names for identification in java.

- It may be class name, method name, variable name. As in below example Car is identifier of class, and X is a identifier of variable.

Example :

class Car{

int x=10;

int y = 20;

}

<u>Note:</u> In java every sentence ends with semicolon (;)

<u>Rules to define Identifiers:</u>

<u>Rule 1:</u> The only allowed characters in java identifiers are

1) a to z

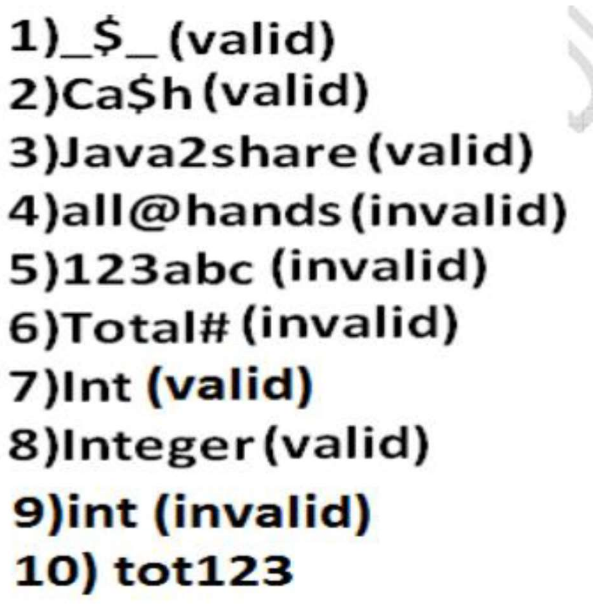
2) A to Z

3) 0 to 9

4) _ (underscore)

5) $

<u>Rule 2:</u> If we are using any other character we will get compile time error.



1)_$_(valid)
2)Ca$h(valid)
3)Java2share(valid)
4)all@hands(invalid)
5)123abc (invalid)
6)Total#(invalid)
7)Int (valid)
8)Integer(valid)
9)int (invalid)
10) tot123

Rule 3: identifiers are not allowed to starts with digit.

Example:

```
Class 1Ram{                    (invalid)

}
```

<u>Rule 4:</u>

- Java identifiers are case sensitive.

- Class name always starts with capital letter.

Example:

```
class Test{
```

String ajay;

}

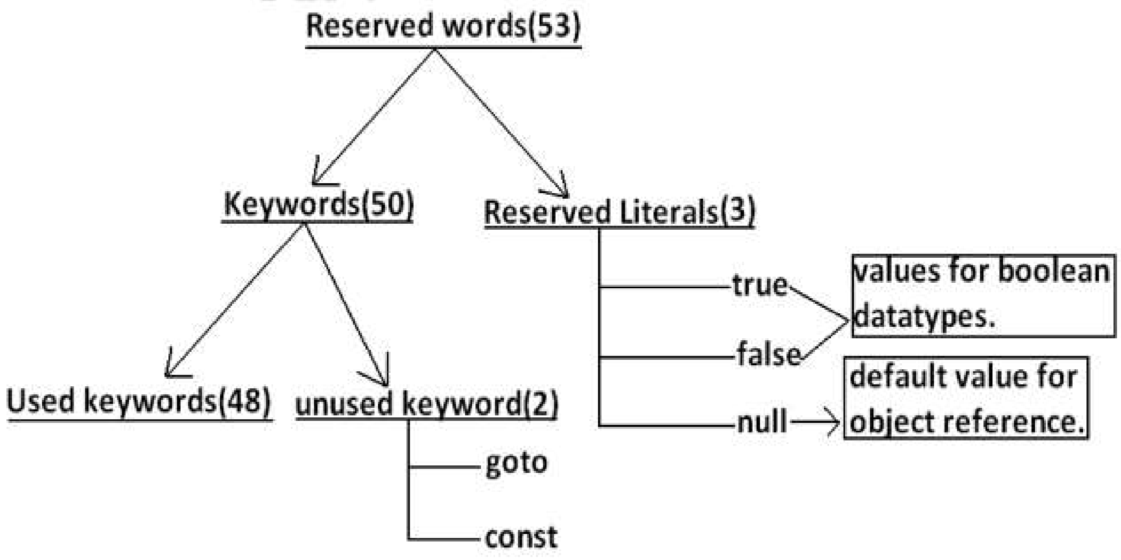
Rule 5: There is no length limit for java identifiers but it is not recommended to take more than 15 lengths.

Rule 6: We can't use reserved words as identifiers

Reserved Words:

In java some words are reserved to associate some functionality or meaning such type of reserved identifiers are called reserved words. This words are predefined by java and cannot used as identifiers.

There are 53 reserved words.



Reserved words for data types: (8)

1) byte

2) short

3) int

4) long

5) float

6) double

7) char

8) Boolean

Reserved words for flow control:(11)

1) if

2) else

3) switch

4) case

5) default

6) for

7) do

8) while

9) break

10) continue

11) return

Keywords for modifiers:(11)

1) public

2) private

3) protected

4) static

5) final

6) abstract

7) synchronized

8) native

9) strictfp(1.2 version)

10) transient

11) volatile

Keywords for exception handling:(6)

1) try

2) catch

3) finally

4) throw

5) throws

6) assert(1.4 version)

Class related keywords:(6)

1) class

2) package

3) import

4) extends

5) implements

6) interface

Object related keywords:(4)

1) new

2) instanceof

3) super

4) this

Java Basic Programming Elements:

1) **Package:**

Packages are group of classes which is used to avoid naming conflicts of same class name. Inside package we need to write our classes.

It is a java folder used to group classes, interfaces and enums.

Ex:

```
package com.software;

public class FirstProgram {

    public static void main(String[] args) {

        int x=10;

        int y=20;
```

```
            //System.out.println("Adding x and y");

            System.out.println(x+y);

        }

}
```

2) **Class:**

3) **Interface:** All these three are java files used to group java data and logic.

4) **Enum:**

5) **Variable:** It’s a named memory location used to store java data.

6) **Method:** It’s a sub block of a class used to implement logic of an object operations.

<u>Basic Steps for Project Creation:</u>

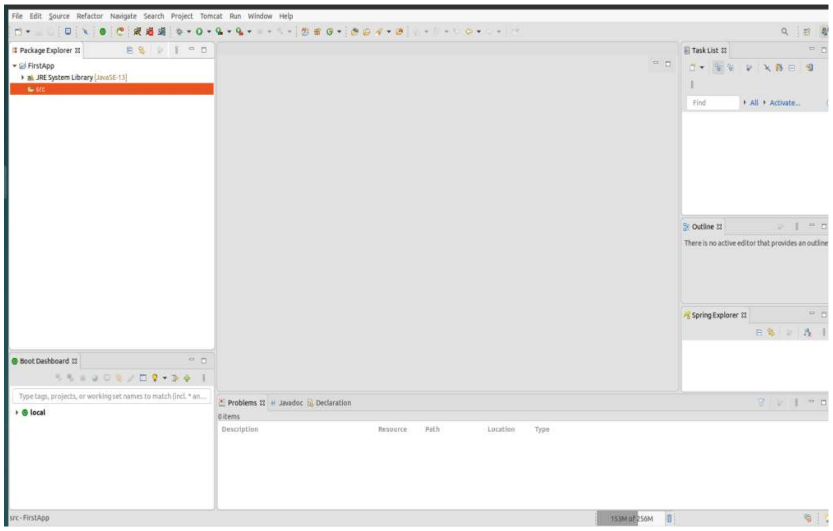
1) Open Eclipse IDE

2) Click on file menu

3) Click on New Option and select other option from the list

4) Search ‘Java Project’ choose it and click on next button

5) Give Project Name and click on finish



Basic Steps for Package Creation:

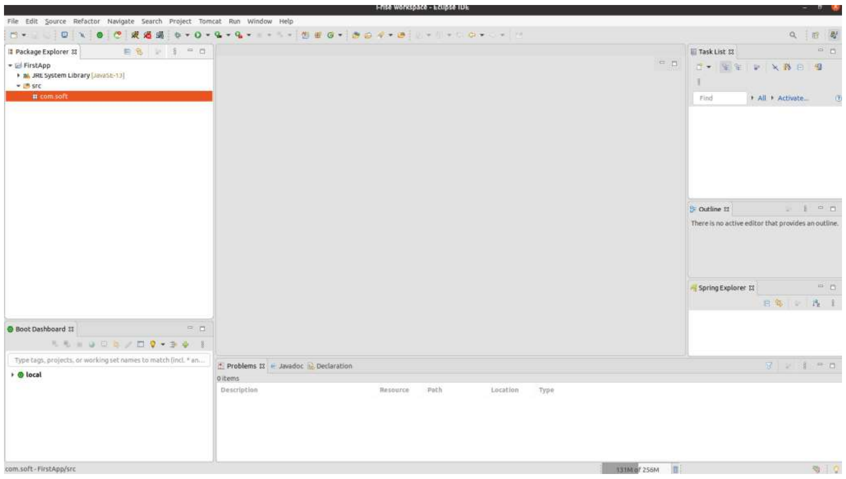
1) Right click on ‘src’ folder

2) Click on New and select other option

3) Search ‘package’ choose it and click on Next button

4) Give package name. (**Try here identifier rule)**

5) Click on finish button



Basic Steps for Class Creation:

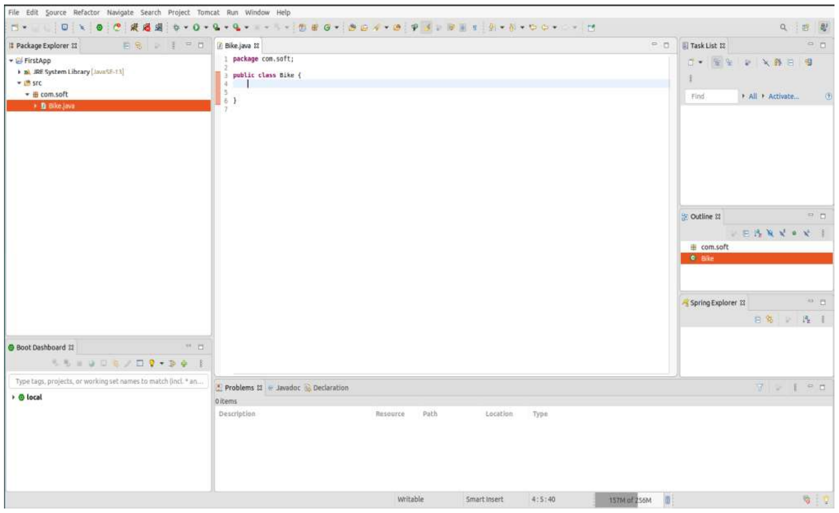
1) Right click on ‘package’ (we already created above).

2) Click on New and select other option

3) Search ‘class’ choose it and click on Next button

4) Give class name.(**Try here identifier rule)**

5) Click on finish button



```
package com.soft;

public class Bike {
	

}
```

```
package com.soft;       // package name

public class Bike {

        int a = 10;         // variable declaration

  public void methodName() {       // method declaration

  }

}
```

Why Java program start with a class?

- The real world object can be represented only using class.
- We can define methods with logic only using class.

Use of main Method:

- The main method is the mediator between java developer and JVM to inform that are the methods should be execution when and which order.

- This method required for execution not for compilation.

- Its calling from JVM automatically.

Example:

```
public class FirstProgram {

    public static void main(String[] args) {

        // Execution start from here.

    }

}
```

Why it is public?:

Because it must be called by JVM from outside of our package.

Why it is static?:

Because it is executes the method logic without object creation.

Why return type void?:

Because we return a value, its sent to jvm which is useless hence main method() return type is void.

Why it is name is main()?:

Because it is executes main logic of program.

Why parameter has String[]?:

To read command line arguments (values) into java application from keyboard.

Why parameter name is args?:

Because we are reading arguments from keyboard the name of the parameter is 'args' we can change its name, but its not recommended.

DataTypes:

- Data type are use to represent the type of data / information.
- Data types are used to stored data temporary in computer through program.
- Data types is something which is give information about –

1) Size of memory location and range of data can be recommended inside that location.

2) Possible legal operation those can be performed on that location.

Example:

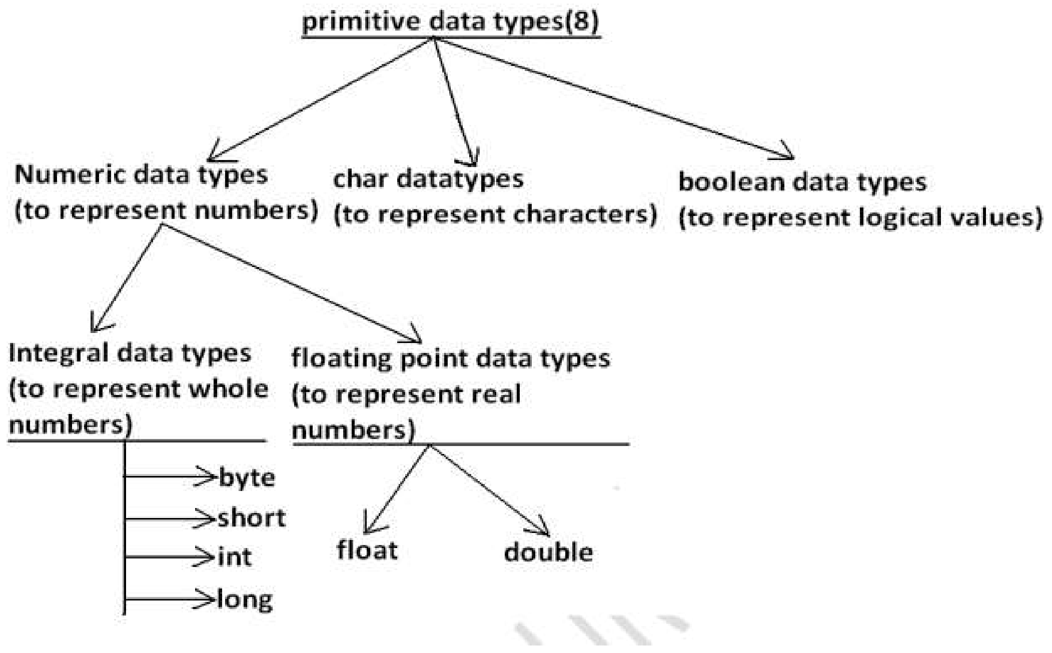
```
int x=10;

boolean b=true;

String name="ABC";

float price= 120.150;

char ch='a';
```

Primitive data types:

- All the primitive data type have fixed size.
- It is used to store single value at a time.
- The primitive data types include byte, short, integer, long, float, double, char, Boolean.

Non-primitive data types:

- The non-primitive data types include Classes, Interfaces, Arrays & String.
- Non Primitive data type don't have fixed size.
- It is used to collect multiple values using primitive types.


primitive data types(8)
Numeric data types (to represent numbers)
char datatypes (to represent characters)
boolean data types (to represent logical values)
Integral data types (to represent whole numbers)
floating point data types (to represent real numbers)
byte
short
int
long
float
double

Primitive Data Type

Ex:

<u>Bytes</u>:(1 byte**)**

byte num1=127;

short(2 bytes)

byte num1=-128;

Integer:(4 bytes)

int x=10;

int y=20;

Float: (4 bytes)

float f=10.5f;

Double: (8 bytes)

double num1=5482037845.2435d;
double num2=10;

Long: (8 bytes)

long num1=10L;
long num2=-10L;

Boolean: (1 bit)

boolean val1=true;

boolean val2=false;

char: (2byte)

- Character in java declare in single quote(‘ ’).

char ch1=’c’;

char ch2= ’s’;

```
package com.soft;
```

```
public class A {


    byte num =120;

    short s = 10001;

    int a = 1000000000;

    long l = 1000000000000002111L;

    char ch='a';

    boolean b = true;

    float f = 10010111111111111111.00f;

    double d =
1258774422222222222222444444444445555555555555554.22574411d;


}
```

Literals in Java

Any constant value which is assigned to the variable are called literals.

Types of literals:

1) Integral literals
2) Floating points literals
3) Character literals
4) String literals

Example:

```
int a= 10;
```

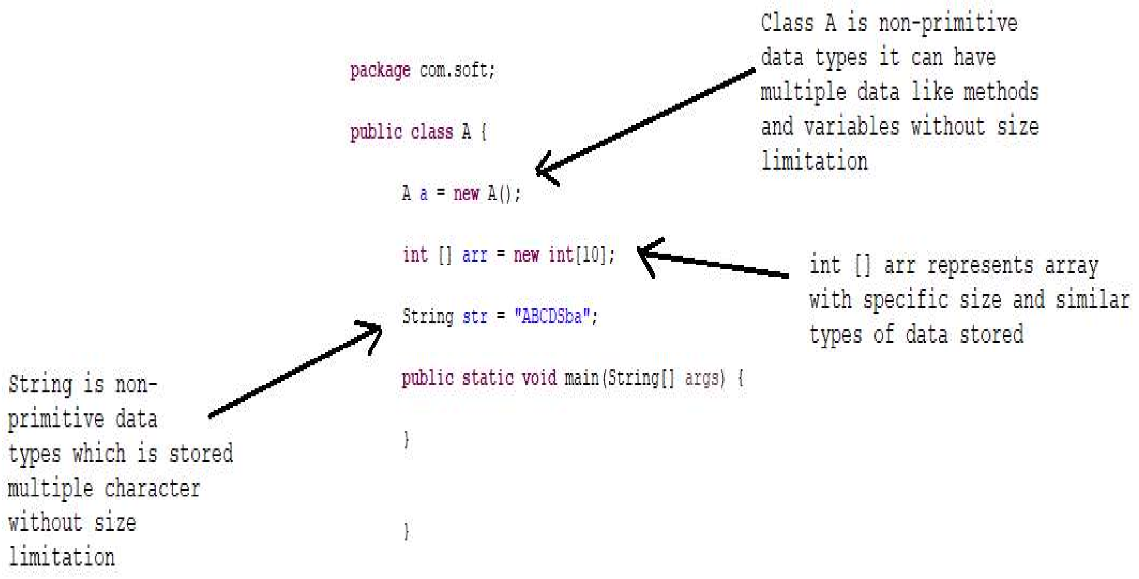
Float f= 10.2;

Char ch = ‘a’;

String str =”abcd”;

Non-primitive data types:

1) Class
2) Interface
3) Array
4) String



```
package com.soft;

public class A {

    A a = new A();

    int [] arr = new int[10];

    String str = "ABCDSba";

    public static void main(String[] args) {

    }

    }
```

Class A is non-primitive data types it can have multiple data like methods and variables without size limitation

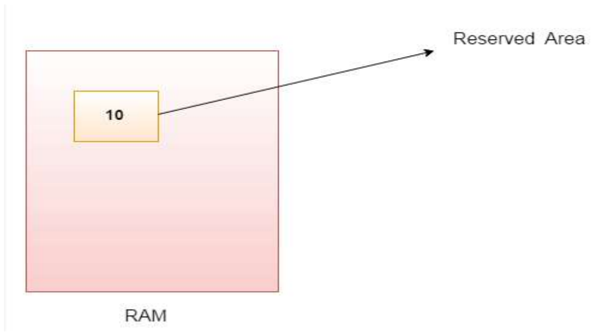
int [] arr represents array with specific size and similar types of data stored

String is non-primitive data types which is stored multiple character without size limitation

```
public class FirstProgram {
        static int a;

        public static void main(String[] args) {

          int b;

           System.out.println(a);     // CE Can’t
Initialized

         System.out.println(b);


        }

    }
```

Variables in Java:

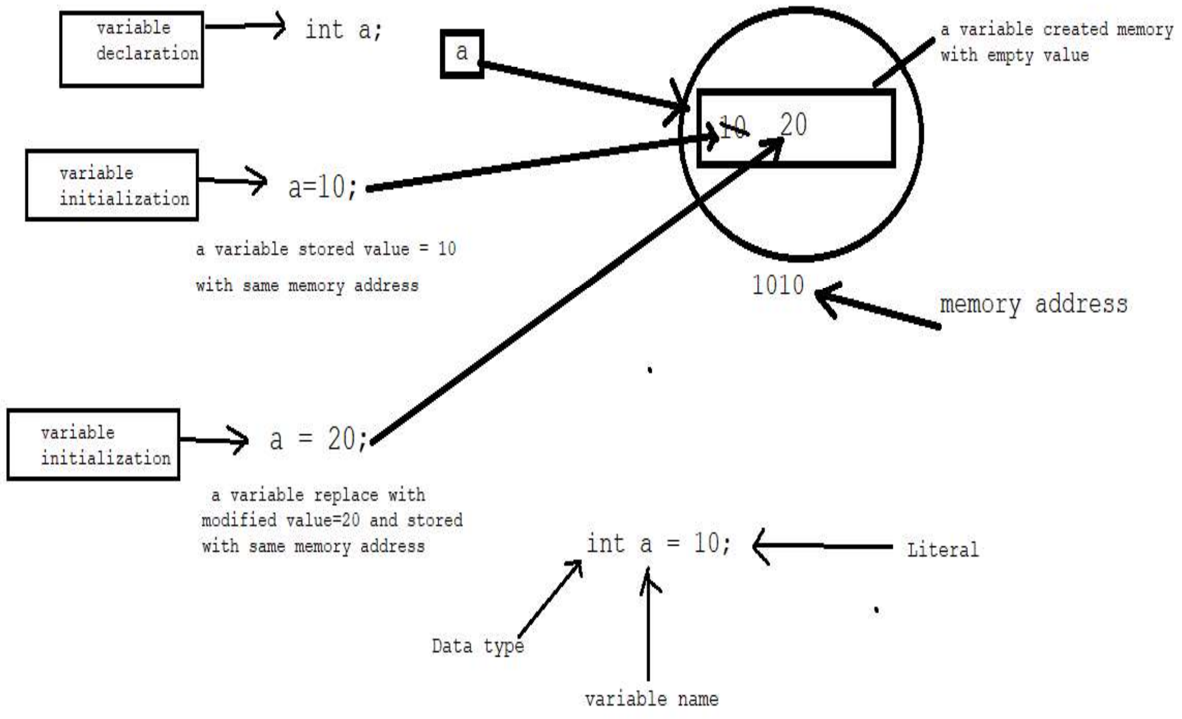
- It is piece of memory which is used to stored data / information. In other words, it is a name of the memory location used to stored data temporarily.
- It is a combination of "vary + able" which means its value can be changed during the execution of the program.
- We should follow identifier rule to declare the variables.



Ex:

Int x=10;

Float f = 10.5f;


variable declaration
int a;
a
a variable created memory with empty value
variable initialization
a=10;
~~10~~ 20
a variable stored value = 10 with same memory address
1010
memory address
variable initialization
a = 20;
a variable replace with modified value=20 and stored with same memory address
int a = 10;
Literal
Data type
variable name

Types of Variables

There are three types of variables in Java:

1) Method level variable (Local Variable)

- The Variable which declare inside method or constructor or block are called as local variable.
- local variable are temporary variable and the variable can't call outside the method or constructor or block.
- The local variable will be created as part of the method execution, hence the scope of the local variable is exactly same as scope of the method .

- A local variable cannot be defined with "static" keyword.

```
package com.soft;
public class A {


    Public static void main (String [] args) {

        int a=10;                    // local variable
        char ch ='a';
        float f =10.5f;


    }



}
```

2) Class level variable (Global variable)

- A variable declared inside the class but outside the body of the method or constructor or block, is called an instance variable or global variable.
- The scope of instance variable is throughout the program.
- i.e. instance Variable can be call anywhere in a program.
- Two types of global variable.

1) Static variable:

The class level variable which has static keyword in its creation statement is called static variable.

Static variable can be access without creation of object.

Memory allocation for static variables happens only once when the class is loaded in the memory, hence it is also known as class level variable.

JVM create memory inside Method/class area for static variable.

Example:

Static int a=10;

```
package com.soft;

public class A {

    static int a =10;      // static variable
which declare as class level
    static char ch ='a';
    float f = 10.5f;

    public static void main(String[] args) {

    int a=10;     // local variable which is de-
clare as method level
    char ch ='a';
    float f =10.5f;


    }



}
```

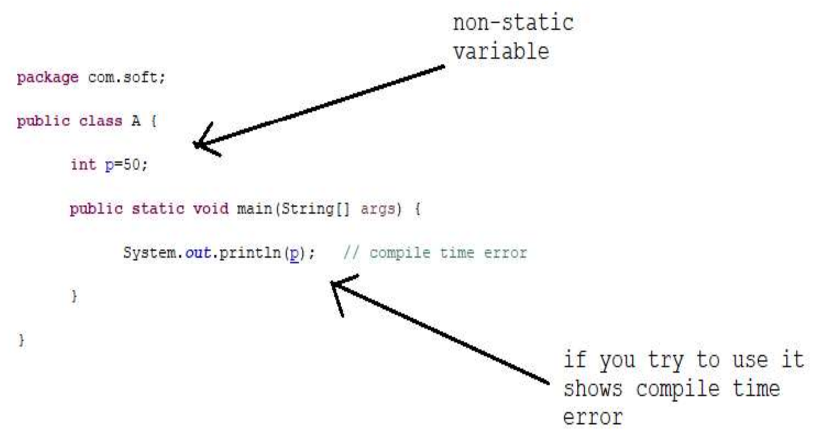
2) Non static variable:

The class level variable which has no static keyword in its creation statement is called non static variable.

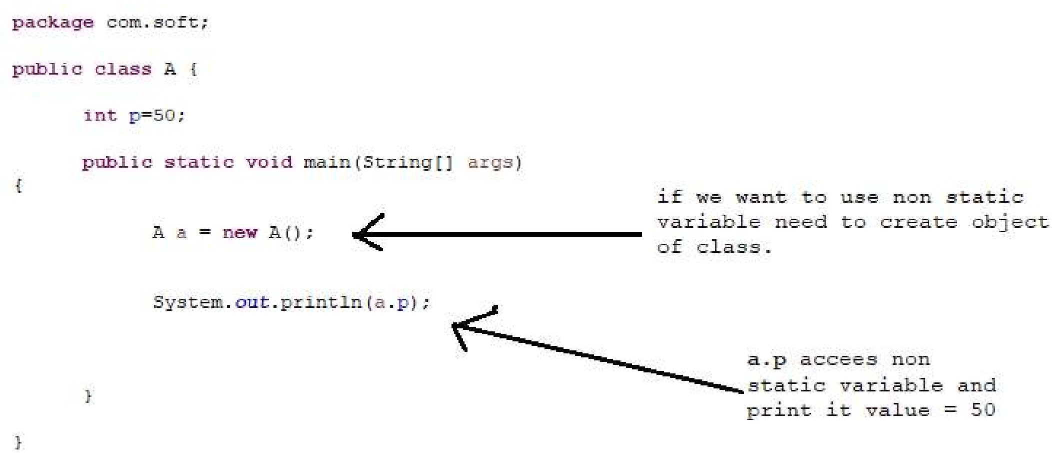
Non static variable can't access without creation object of class i.e. means we need to create instance of a class to access it.

Ex:

```
public class A
{
   static int m=100;//static variable
   int x =20;
   void method()
   {
      int n=90;//local variable
   }
   public static void main(String args[])
   {
      A a = new A();
       System.out.println(a.x);
   }
}//end of class
```



```
package com.soft;

public class A {

    int p=50;

    public static void main(String[] args) {

        System.out.println(p);   // compile time error

    }

}
```



```
package com.soft;

public class A {

      int p=50;

      public static void main(String[] args)
{

            A a = new A();

            System.out.println(a.p);


      }

}
```

if we want to use non static variable need to create object of class.

a.p accees non static variable and print it value = 50

Access Specifiers:

- Access Specifier are use to define the scope of element declare inside class.

- The keyword which define accessibility permissions are called access specifier.

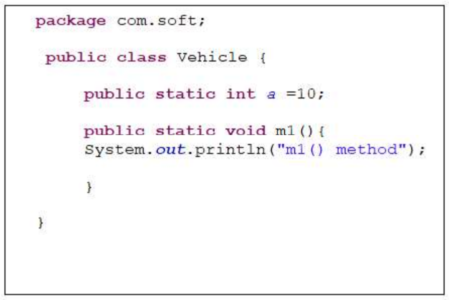
- In java we can change the access level of variable, constructors, methods, and Class by applying the access modifier on it.

There are 4 types of Access Specifier:

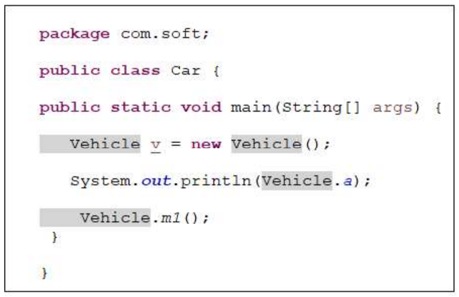
1) Public
2) Default
3) Protected
4) Private

1) Public Access Specifier:

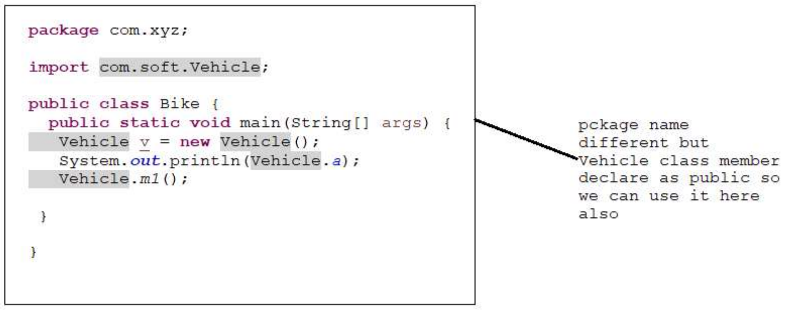
- The scope of public Access Specifier is throughout the project i.e. the element declare with public Access Specifier can be call in any class of same project.
- Public is a keyword in java.



```
package com.soft;

 public class Vehicle {

     public static int a =10;

     public static void m1(){
     System.out.println("m1() method");

     }

}
```



```
package com.soft;

public class Car {

public static void main(String[] args) {

   Vehicle v = new Vehicle();

   System.out.println(Vehicle.a);

    Vehicle.m1();
 }

}
```



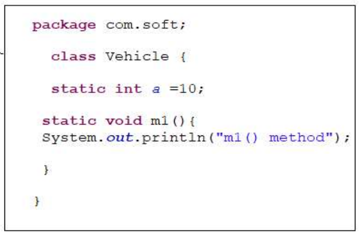
```
package com.xyz;

import com.soft.Vehicle;

public class Bike {
  public static void main(String[] args) {
   Vehicle v = new Vehicle();
   System.out.println(Vehicle.a);
   Vehicle.m1();

 }

}
```

pckage name different but Vehicle class member declare as public so we can use it here also

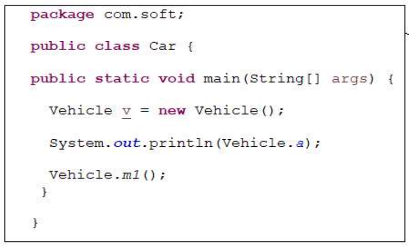
2) Default Access Specifier:

- The Scope of default Access Specifier is within package only.
- i.e. The element declare with default Access Specifier can be call in all the other classes within same package.
- There is No Keyword for default Access Specifier.

default keyword not use to declare access specifier

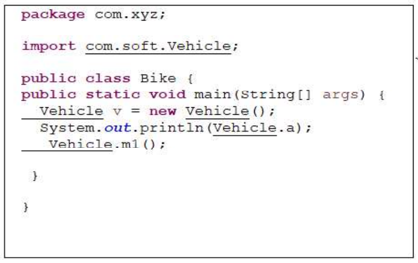


```
package com.soft;

  class Vehicle {

  static int a =10;

 static void m1(){
 System.out.println("m1() method");

 }

}
```



```
package com.soft;

public class Car {

public static void main(String[] args) {

  Vehicle v = new Vehicle();

  System.out.println(Vehicle.a);

  Vehicle.m1();
 }

}
```

No compile tim error,Vehicle class memeber declare as default



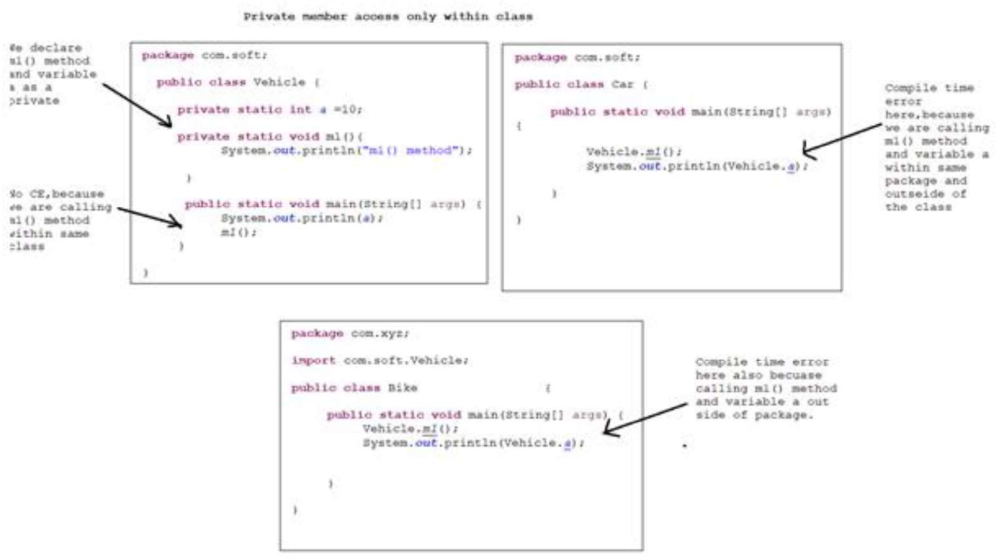
```
package com.xyz;

import com.soft.Vehicle;

public class Bike {
public static void main(String[] args) {
  Vehicle v = new Vehicle();
  System.out.println(Vehicle.a);
   Vehicle.m1();

 }

}
```

Compile time error ,package name different here and Vehicle Class memeber declare as default, so default access specifier can not use outside of the package

3) Private Access Specifier:

- The Scope of private Access Specifier is within class only.
- The element declare with private Access Specifier cannot be call outside the class.
- Private is keyword is use to denote private Access Specifier.
- It is most restrictive access specifier.

Note: private and protected access specifier not use to declare for Class.



Private member access only within class

We declare m1() method and variable a as a private

No CE,because we are calling m1() method within same class

```
package com.soft;

  public class Vehicle {

     private static int a =10;

     private static void m1(){
           System.out.println("m1() method");

      }

      public static void main(String[] args) {
           System.out.println(a);
           m1();
     }

}
```

```
package com.soft;

public class Car {

     public static void main(String[] args)
{

          Vehicle.m1();
          System.out.println(Vehicle.a);

     }

}
```

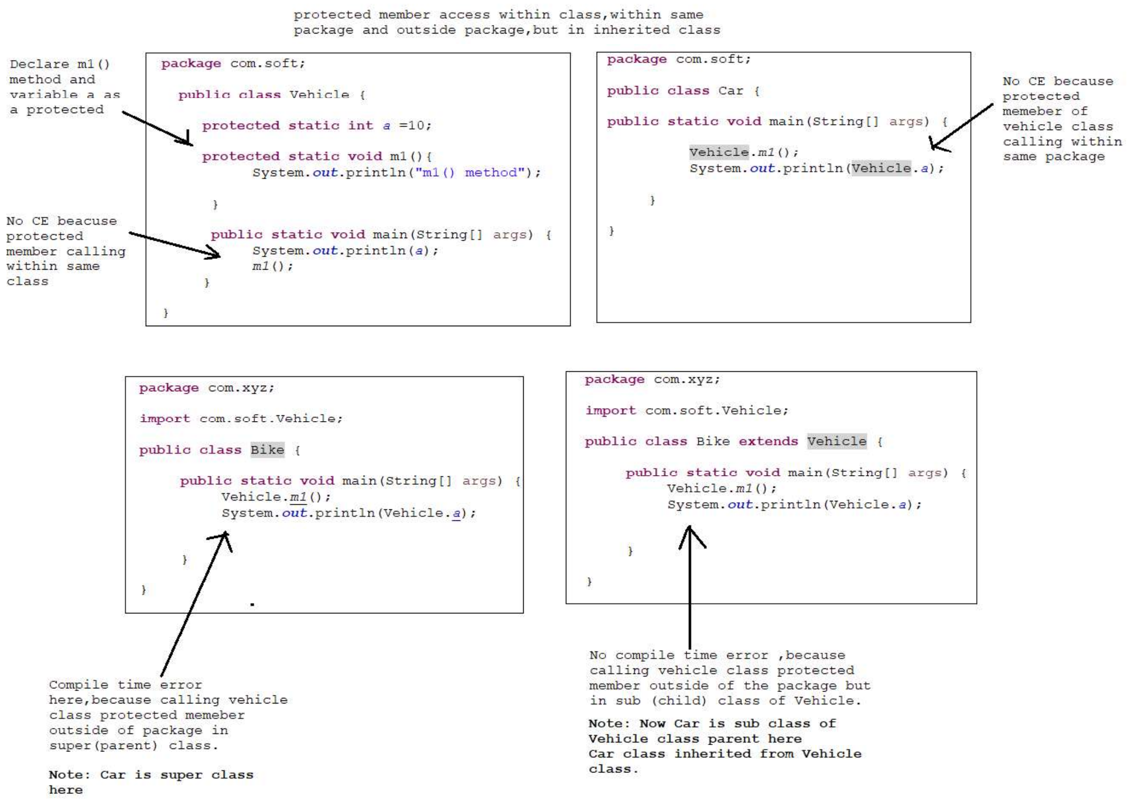
Compile time error here,because we are calling m1() method and variable a within same package and outseide of the class

```
package com.xyz;

import com.soft.Vehicle;

public class Bike                 {

     public static void main(String[] args) {
          Vehicle.m1();
          System.out.println(Vehicle.a);


     }

}
```

Compile time error here also becuase calling m1() method and variable a out side of package.

4) Protected Access Specifier:

- The Scope of protected Access Specifier is same as default Access Specifier i.e. within the package only.
- But the difference is the protected properties can be call outside the packages on one condition i.e. inheritance (Super class – class A, Sub Class – Class B)
- It creates protected members and those can be accessible within a package from all classes, but outside package only in subclass that too only by subclass object.
- Protected is a keyword use to denote protected Access specifier.



protected member access within class,within same package and outside package,but in inherited class

Declare m1() method and variable a as a protected

No CE beacuse protected member calling within same class

```
package com.soft;

  public class Vehicle {

     protected static int a =10;

     protected static void m1(){
           System.out.println("m1() method");

      }

      public static void main(String[] args) {
           System.out.println(a);
           m1();
     }

}
```

```
package com.soft;

public class Car {

public static void main(String[] args) {

           Vehicle.m1();
           System.out.println(Vehicle.a);

     }

}
```

No CE because protected memeber of vehicle class calling within same package

```
package com.xyz;

import com.soft.Vehicle;

public class Bike {

     public static void main(String[] args) {
           Vehicle.m1();
           System.out.println(Vehicle.a);


     }

}
```

Compile time error here,because calling vehicle class protected memeber outside of package in super(parent) class.

Note: Car is super class here

```
package com.xyz;

import com.soft.Vehicle;

public class Bike extends Vehicle {

     public static void main(String[] args) {
           Vehicle.m1();
           System.out.println(Vehicle.a);


     }

}
```

No compile time error ,because calling vehicle class protected member outside of the package but in sub (child) class of Vehicle.

Note: Now Car is sub class of Vehicle class parent here Car class inherited from Vehicle class.

What is method in java:

A **method** is a block of code or collection of statements or a set of code grouped together to perform a certain task or operation which means used for implementing logic of an object operation. It provides the reusability of code. We can also easily modify code using **methods** , we will learn **what is a method in Java, types of methods, method declaration,** and **how to call a method in Java**.

Types of Method:

- Pre-Define Method
- User Define Method

- **Pre-Define Method or System Define Method**