To run java application in our pc we must have an installed JDK.

JDK is contains the tools that are used to develop the java applications.

JRE provides the java application run time environment. It is simply the implementation for JVM.

JVM performs:

**Loads code**  
**Verifies code**  
**Executes code**  
**Provides runtime environment**

**Compiler:** A compiler is used to convert user level language to machine level language. Because a computer can only understand machine level language. Compiler executes the total program at once. Compiler type languages are C, C++, Java and etc. Here code execution is faster, and it is heavy programming.

Generates intermediate object code which further requires linking, hence requires more memory.

It generates the error message only after scanning the whole program. Hence debugging is comparatively hard.



**Interpreter**: Interpreter executes code line by line. Since it executes line by line it is less weight to the machine but take more time to execute.

No intermediate object code is generated, hence are memory efficient.

Continues translating the program until the first error is met, in which case it stops. Hence debugging is easy.

Programming language like Python, Ruby use interpreters.

**Class** is template with methods properties, events. Instances of the class Class represent classes and interfaces in a running Java application. An enum is a kind of class and an annotation is a kind of interface. Every array also belongs to a class that is reflected as a Class object that is shared by all arrays with the same element type and number of dimensions. The primitive Java types (boolean, byte, char, short, int, long, float, and double), and the keyword void are also represented as Class objects.

Class has no public constructor. Instead Class objects are constructed automatically by the Java Virtual Machine as classes are loaded and by calls to the defineClass method in the class loader.

Since java is an object oriented language it is used to create class that acts as a instance of objects.

**Object:** An object contains both methods and data are used to show its state and behavior. It stores its state in fields called variables and behavior through containing methods.  Methods operate on an object's internal state and serve as the primary mechanism for object-to-object communication. Hiding internal state and requiring all interaction to be performed through an object's methods is known as data encapsulation — a fundamental principle of object-oriented programming. With object oriented programming we are able to achieve

**Modularity:** The source code for an object can be written and maintained independently of the source code for other objects. Once created, an object can be easily passed around inside the system.

**Information-hiding:** By interacting only with an object's methods, the details of its internal implementation remain hidden from the outside world.

**Code re-use:** If an object already exists (perhaps written by another software developer), you can use that object in your program. This allows specialists to implement/test/debug complex, task-specific objects, which you can then trust to run in your own code.

**Pluggability and debugging ease:** If a particular object turns out to be problematic, you can simply remove it from your application and plug in a different object as its replacement. This is analogous to fixing mechanical problems in the real world. If a bolt breaks, you replace it, not the entire machine.

Variables: Java works mainly on 8 different kinds of variable data types. In which primitive data types are byte(1), short(2bytes), int(4b), float(4), double(8), long(8), character(2), Boolean(1), and one non primitive data type that is String.

\*\*There is no operator overloading in Java! The operator + is only defined for strings, you will never see it with other objects, only primitives.

\*\*Every comparison operator in java returns the type Boolean.

e write methods in class. We write logic(our test case steps) in side method. it will contain two parts:

method signature will contain:

**Modifiers**—such as public, private, and others you will learn about later.

The return type—the data type of the value returned by the method, or void if the method does not return a value.

The method name—

The parameter list in parenthesis—If there are no parameters, you must use empty parentheses.

**Public static void main(String[] args){**

**}**

There where the program execution will start. Without main method we can not run the project and not able to get output. This is also called as the entry point for all java applications.

A Java program starts by executing the main method of some class. You can choose the name of the class to execute, but not the name of the method. The method must always be called main.

Methods:

A method basically contains **method signature** and **method body**. Method body is nothing but every content that is placed between the braces {} of the that method.

Method overring is the concept that is performed to use the supper class properties in the child class. In other cases, it is not acceptable like in general static method cases.

Ex: If a method has same name in common static cases it won’t be accepted in all the cases.

\*\*If there are some methods in the class and we want to use them in the other classes we have to use them by using the className.mehtodname();

This works only to those classes that are belong to all the packages of that project and it should defined as **public,** and as well as **static**.

If a method is **non-static** then we can not access it across static methods. To get access to those methods we have call them by using an **instance** variable. And static methods should be accessed in **static way only**

**Private** is limited to local use only, so may method that is declared as private we can only access it across the local class.

**Default, protected** is limited to package we cannot access default classes out of package.

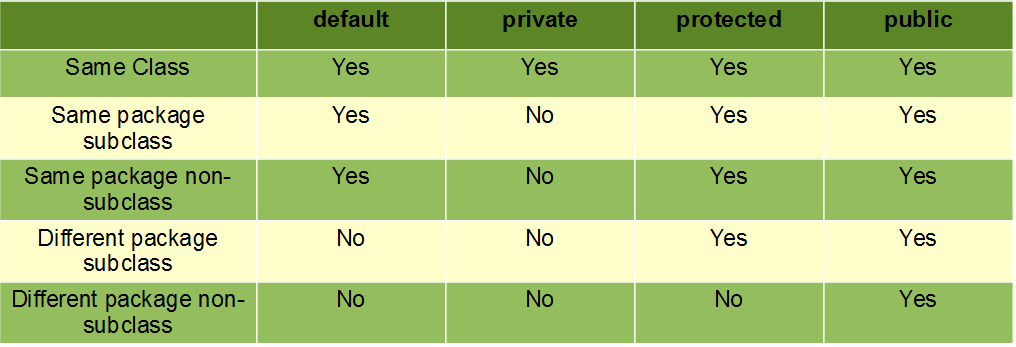
\*\*Void methods cannot return a value.

\*\*While importing a class from different package we use the:

Projectname. Classthatwewant;

Ex: saibersysJava.creatingMethods;

Public, private, protected, and default. Access modifiers.



**Objects:** In java objects do the all kind of work. If basically contains data and methods that we use to perform the work.

**Class:** Class is basically containing instructions how objects can be created in and define how they behave. Class is a specification or blueprint or design to say how a class should behave in the application.

To use the behavior of the class as object from we must have a constructor. By default, java provides a zero-argument constructor without any return type. To have constructor we must define it with the name as same as the className. It looks as a special method.

By default all the variable in java assigned to “Null” or 0

Human name = new Human();

Where as name is called as instance variable. It points to actual object. It is also called as reference variable.

Methods are once that do all the work in java.

**Stack memory**: When the main method starts executing it starts crating memory in the stack. Along with the variables that it has. Similarly, each and every method and its body get stack memory one over the other.

**Heap Memory:** If an object is created that objects get to be assigned a memory slot in heap memory. This memory is referenced by the variable memory that is crated under main memory stack.

**Garbage collection:** It is a process that runs in the heap memory. It looks for objects that don’t have memory relation with the reference variables and clears that memory for future use.