Method Overloading Resolution :

* While working with method overloading (same parameters), compiler will select the method based on this priorities.
* While executing the method, Compiler will provide the priority by using the following rule.

1) EXACT MATCH rule
   Example:
   Accept(5);   → Here we are passing the value 5 which is an integer value so, the first priority Compiler will provide to **int type**

2) Most specific super can
   Higher → Bjava   That is the more specific type
   float → long
   long → int
   int → short   (There is no relation between short & char.)
   short → byte

   So the conclusion is :
   byte → short → int → long → float → double
   char → int → long → float → double

3) After (Widening) → Autoboxing → var Args
   While working with Widening, Autoboxing and var Args the compiler will give the priority as the following order.
   1) Widening
   2) autoboxing
   3) Var Args

(Programs on method Overloading)

```
MO-1.java
void main()
{
    accept(5);
}
public void accept(double d)
{
    S.o.println("double");
}
public void accept(float f)
{
    S.o.println("float");
}
```

Here , Here float will be executed because float is the most specific type.

```
MO-2.java
void main()
{
    accept(5);
}
public void accept(int i)
{
    S.o.println("int");
}
public void accept(long l)
{
    S.o.println("long");
}
```

Here it is int type so int will be executed.

```
MO-3.java
void main()
{
    accept('a');
}
public void accept(int i)
{
    S.o.println("int");
}
public void accept(char c)
{
    S.o.println("char");
}
```

char will be executed becoz char is more specific type.

```
MO-4.java
void main()
{
    accept(5);
}
public void accept(short s)
{
    S.o.println("short");
}
public void accept(char c)
{
    S.o.println("char");
}
```

Here we will get compilation error because there is no relation between char and short based on the specific type rule.

```
MO-05.java
void main()
{
    accept();
}
public void accept(double... d)
{
    S.o.println("dou");
}
public void accept(int... i)
{
    S.o.println("int");
}
```

Here bata will be executed because bata is the specific type.

```
MO-06.java
void main()
{
    accept();
}
public void accept(double ...d)
{
    S.o.println("double");
}
public void accept(long ...l)
{
    S.o.println("long");
}
```

Here long will be executed because long is the more specific type.

```
MO-07.java
void main()
{
    accept(5);
}
public void accept(int i)
{
    S.o.println("int");
}
public void accept(long l)
{
    S.o.println("long");
}
```

Here , Here int will be executed because int is the exact match.

```
MO-08.java
void main()
{
    accept(5);
}
public void accept(long l)
{
    S.o.println("Widening");
}
public void accept(Integer I)
{
    S.o.println("autoboxing");
}
```

Here widening is having more priority.

```
MO-09.java
void main()
{
    accept(5);
}
public void accept(int ...i)
{
    S.o.println("Var args");
}
public void accept(Integer I)
{
    S.o.println("autoboxing");
}
```

Here Autoboxing will be executed.

```
MO-10.java
void main()
{
    accept(5);
}
public void accept(int ...i)
{
    S.o.println("Var args");
}
public void accept(int i)
{
    S.o.println("int");
}
```

Here int will be executed.

```
MO-11.java
void main()
{
    accept(5);
}
public void accept(int ...i)
{
    S.o.println("Var args");
}
public void accept(int i)
{
    S.o.println("int");
}
```

Here we will get compilation error.

```
MO-12.java
void main()
{
    accept(5);
}
public void accept(int ...i)
{
    S.o.println("Var args");
}
public void accept(Integer x, Integer y)
{
    S.o.println("Widening");
}
```

Here Widening will be executed.

```
MO-13.java
void main()
{
    accept(5);
}
public void accept(int i, int j)
{
    S.o.println("i+ - j+g");
}
public void accept(long x, int z)
{
    S.o.println("long - int");
}
```

we can get compilation error.

```
MO-xx.java
void main()
{
    accept(5);
}
public void accept(double x)
{
    S.o.println("Double");
}
public void accept(long l)
{
    S.o.println("Long - ");
}
```

Compilation error.

What is MAIN and the class ?

* Every Java program must be executed by using at least one class.
* Every Java (JVL files) is faulty, then if we don't write a class then **Java compiler will automatically add a class, which is based on the Java file name known as "Compact class & file".**
* As a developer we can write a class in our Java application so, Java compiler will not generate any class.

Int class :
* It stands for "Business Logic Java".
* We should always write our **business logic** model int class by using methods.
* This Int class doesn't contain main method. **int will never contain main method**
  Example:
  ```
  public class BLC
  {
      //This code should not contain main method
  }
  ```

RJC Class:
* It stands for "Executable Logic Class".
* Our execution of the program will start from ELC class because it's once contain main method.
  Example:
  ```
  public class ELC
  {
      public static void main(String [ ] args) //This main method is static  method
      {
      }
  }
  ```

Note : Before Java AND the main method was static method so PVM was calling the method with the help of class name on the other hand from JVM. If's main method is non-static then main method can't be called by PVM with class name and that's not a non-static method

What is ECLIPSE IDE :

IDE stands for "Integrated Development Environment". By using eclipse IDE, we can develop, compile and execute our Java program in a single window.

The main purpose of eclipse IDE to reduce the development time, since the development time will be reduced then automatically the cost of the project will be reduced.

How to create a Project in Eclipse IDE :
File → new → Project OR Java Project → Provide the name for the project (Applic-to_xyz) → Finish

What is a Package & Area :

A package is nothing but folder in windows. It is used to arrange the classes and interfaces into a particular group.

If we arrange our Java classes into a particular group by using packages (folders) then we will get the following two advantages :
1) Fast searching is possible.
2) Name can be reusable.

Types of Packages :
1) Predefined OR Built-in package : The packages which are created by Java software people for arranging the programs are called predefined package.
   Example : java.lang, Java.util, Java.io, Java.net, Java.sol etc etc
2) UserDefined Package OR Custom package : The packages which are created by user for arranging the user defined programs are called user defined package.
   Example:
   com.xyz.loans
   com.xyz.adm
   com.xyz.emm1_shopping

Can we write multiple public classes in a single .java file:

No, in a single java file, We are always only one public class, remaining all the classes must be non public.

This public state must be our file name in java.


```
package com.city.blc;
public class Square
{
    public static void getSquare(int side)
    {
        S.o.print("Area of square is :"+(side* side));
    }
}
```

```
package com.city.elc;
import com.city.Square;
public class ELC
{
    public static void main(String[] args)
    {
        Square.getSquare(5);
    }
}
```

Structure of Java Program :
1st → Package statement
2nd → Import statement
3rd → public class BLC/ELC { }