Command Line Argument:

The arguments which are passing from command prompt are called command line argument. With these command line arguments JVM will create an array and by passing that array as argument JVM will call main() method. The main objective of command line arguments is we can customize behavior of the main() method.

Ex: java test A B C ↵ args[0] = A args[1] = B args[2] = C args.length => 3

```
class test
{
public static void main(String args[])
{
int a = Integer.parseInt(args[0]);
System.out.println("the square of "+ a+" is : "+(a * a));
}
}
```

```
C:\Users\Atish kumar sahu>cd desktop

C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test 5
the square of 5 is : 25

C:\Users\Atish kumar sahu\desktop>java test 6
the square of 6 is : 36

C:\Users\Atish kumar sahu\desktop>
```

```
class test
{
public static void main(String args[])
{
for(int i = 0 ; i <= args.length ; i++)
{
System.out.println(args[i]);
}
}
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test A B C D E
A
B
C
D
E
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
        at test.main(test.java:7)

C:\Users\Atish kumar sahu\desktop>
```

```
class test
{
public static void main(String args[])
{
for(int i = 0 ; i < args.length ; i++)
{
System.out.println(args[i]);
}
}
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test A B C D E F
A
B
C
D
E
F

C:\Users\Atish kumar sahu\desktop>
```

IF WE REPLACE <= WITH < THEN WE WON'T GET ANY RUNTIME EXCEPTION.

```
class test
{
public static void main(String args[])
{

String[] argh = {"x", "y", "z"};
args=argh;

for(String S : args)
{
System.out.println(S);
}
}
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test A B C D
x
y
z

C:\Users\Atish kumar sahu\desktop>java test A B C
x
y
z

C:\Users\Atish kumar sahu\desktop>java test A B
x
y
z

C:\Users\Atish kumar sahu\desktop>java test A
x
y
z

C:\Users\Atish kumar sahu\desktop>java test
x
y
z
```

```
class test
{
public static void main(String args[])
{

String[] argh = {"x", "y", "z"};

for(String S : args)
{
System.out.println(S);
}
}
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test A B C D
A
B
C
D

C:\Users\Atish kumar sahu\desktop>java test A B C
A
B
C

C:\Users\Atish kumar sahu\desktop>java test A B
A
B

C:\Users\Atish kumar sahu\desktop>java test A
A

C:\Users\Atish kumar sahu\desktop>java test

C:\Users\Atish kumar sahu\desktop>
```

```
class test
{
public static void main(String args[])
{
System.out.println(args[0] + args[1]);
}
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test 10 20
1020

C:\Users\Atish kumar sahu\desktop>
```

WITH IN MAIN() METHOD COMMAND LINE ARGUMENTS ARE AVAILABLE IN STRING FORM SO IT CONCATENATE IN ABOVE EXAMPLE.

```
class test
{
public static void main(String args[])
{
System.out.println(args[0]);
}
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test note book
note
```

```
C:\Users\Atish kumar sahu\desktop>java test "ATISH KUMAR SAHU"
ATISH KUMAR SAHU

C:\Users\Atish kumar sahu\desktop>
```

USUALLY SPACE ITSELF IS THE SEPARATOR BETWEEN COMMAND LINE ARGUMENTS IF OUR COMMAND LINE ARGUMENT ITSELF CONTAINS A SPACE THEN WE HAVE TO ENCLOSE THAT COMMAND LINE ARGUMENT WITH IN DOUBLE QUOTES(“”).

public static void main(String args[]){ }:

Whether class contains main() method or not and whether main() method is declared according to requirement or not these things won't be checked by compiler. At runtime JVM is responsible to check these things. At runtime if JVM is unable to find required main() method then we will get runtime exception => NoSuchMethodError:main

```
class test
{
}
```

```
C:\Users\Atish kumar sahu>cd desktop

C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test
Error: Main method not found in class test, please define the main method as:
   public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application

C:\Users\Atish kumar sahu\desktop>
```

public static void main(String[] args):

public = to call by JVM from anywhere

static = without existing object also JVM has to call this method and main() method no way related to any object

void = main() method won't return anything to JVM

main = this the name which is configured inside JVM source code

String[] args = command line arguments

public static void main(String args[]) syntax is very strict if we perform any change we will get runtime exception saying NoSuchMethodError:main.

Even through the above syntax is very strict the following changes are acceptable.

the order of modifiers is not important that is instead of "public static" we can take "static public" also.

we can declare "String[]" in any acceptable form.

main(String[] args)

main(String []args)

main(String args[])

instead of 'args' we can take any valid java identifier.

we can replace String[] with var arg parameter = main(String[] args) ⍰ main(String... args)

we can declare main() method with the following modifiers also "final" "synchronized" "strictfp".

```
class test
{
final static synchronized strictfp public void main(String... args)
{
System.out.println("ACCEPTABLE MAIN METHOD");
}
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test
ACCEPTABLE MAIN METHOD

C:\Users\Atish kumar sahu\desktop>
```

Which of the following are valid main() method declarations?

1. public static void Main(String[] args)

2. public static int main(String[] args)

3. public static void main(String args)

4. public final synchronized strictfp void main(String[] args)

5. public static final synchronized strictfp void main(String[] args) correct method

6. public static void main(String… args) correct method

In which of the above cases we will get compile time error?

We won't get compile time error anywhere but at runtime we will get run time exception in case 1,2,3,4.

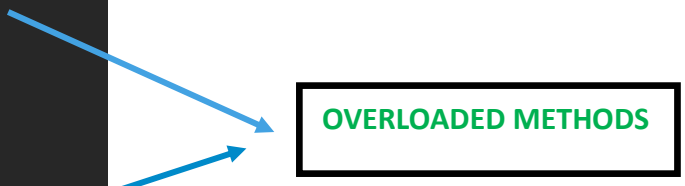
CASE-1: **********

Overloading of the main() method is possible but JVM will always call string[] argument main() method only. The other overloaded method we have to call explicitly then it will be executed just a normal method call.

```
class test
{
//method overloading
public static void main(String[] args)
{
System.out.println("String[]");
}

public static void main(int[] args)
{
System.out.println("int[]");
}

}
```




```
C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test
String[]

C:\Users\Atish kumar sahu\desktop>
```

CASE – 2:

Inheritance concept applicable for the main() method. Hence while executing child class if child class doesn't contain main() method then parent class main() method will be executed.

```
class test
{

public static void main(String args[])
{
System.out.println("PARENT MAIN");
}

}
class test1 extends test
{

}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test
PARENT MAIN

C:\Users\Atish kumar sahu\desktop>java test1
PARENT MAIN

C:\Users\Atish kumar sahu\desktop>
```

Case – 3:

It seems overriding concept applicable for main method but it is not overriding it is method hiding because in both classes methods are in static form.

```
/*method overriding because of
both methods are static it becomes method hiding*/

class test
{
public static void main(String args[])
{
System.out.println("PARENT MAIN");
}
}

class test1 extends test
{
public static void main(String args[])
{
System.out.println("CHILD MAIN");
}
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test
PARENT MAIN

C:\Users\Atish kumar sahu\desktop>java test1
CHILD MAIN

C:\Users\Atish kumar sahu\desktop>
```

FOR MAIN() METHOD INHERITANCE AND OVERLOADING CONCEPTS ARE APPLICABLE BUT OVERRIDING CONCEPT IS NOT APPLICABLE INSTEAD OF OVERRIDING METHOD HIDING CONCEPT IS APPLICABLE.

Java 1.7 version enhancement with respect to main() method:

CASE-1: ********

Until java 1.6 version if the class doesn't contain main() method then we will get runtime exception saying NoSuchMethodError. But from java 1.7 version onwards instead of NoSuchMethodError we will get more meaningful error information.

```
class test
{

}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test
Error: Main method not found in class test, please define the main method as:
   public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application

C:\Users\Atish kumar sahu\desktop>
```

CASE – 2:

From java 1.7 version onwards to run a java program main method is mandatory. Hence even through class contains static blocks they won't be executed if the class doesn't contain main() method.

```
class test
{

static
{
System.out.println("STATIC BLOCK");
}

}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test
Error: Main method not found in class test, please define the main method as:
   public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application

C:\Users\Atish kumar sahu\desktop>
```

1.6 version	1.7 version
javac Test.java	javac Test.java
java Test	java Test
output: static block RE: NoSuchMethodError: main	Error: main method not found in class

```
class test
{

static
{
System.out.println("STATIC BLOCK");
System.exit(0);
}

}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test
Error: Main method not found in class test, please define the main method as:
   public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application

C:\Users\Atish kumar sahu\desktop>
```

1.6 version	1.7 version
javac Test.java	javac Test.java
java Test	java Test
output: static block RE: NoSuchMethodError: main	Error: main method not found in class

```
class test
{

static
{
System.out.println("STATIC BLOCK");
}

public static void main(String args[])
{
System.out.println("MAIN METHOD");
}

}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test
STATIC BLOCK
MAIN METHOD

C:\Users\Atish kumar sahu\desktop>
```

IF THE CLASS CONTAINS MAIN() METHOD WHETHER IT IS VERSION 1.6 OR VERSION 1.7 THERE IS NO CHANGE IN EXECUTION SEQUENCE.

1.6 version	1.7 version
javac Test.java	javac Test.java
java Test	java Test
output: static block method	output: static block main method

Q. WITH OUT WRITING MAIN() METHOD IS IT POSSIBLE TO PRINT SOME STATEMENTS TO THE CONSOLE?

ANS. Yes we can print by using static block. But this rule is applicable until 1.6 version from 1.7 version onwards main() method is mandatory to print some statements to the console.