

### Understanding Java main():

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

**public** ⇒ Because it should be available to all the classes which were present in our project

**static** ⇒ Static methods can be called directly with the help of its respective class name no need of creating an object. As main() is a static method whenever we are running our Java program from the command Java generated.classFileName JVM is going to take the class name and its respective main() will be getting executed

**void** ⇒ Because every Java program execution should start from main() and ends with main(), if main() is having any other return type except void our program execution will not be completed with main().

**main** ⇒ It is JUST a name.

**String [] args** ⇒ Java main method accepts a “single” argument of **type** String array. This is also called as java command line arguments.

## Understanding java main() method

- The Syntax of main method is ==> **public static void main(String []args)**

**public** : Anything declared as public can be accessed from anywhere, main method should be available for other classes in the project. So main method “has” to be public

**static** : Static methods can be called directly with out creating a class object. So when java execution starts, JVM will be able to start the main method.

**void**: A java program execution starts from main() method and ends with main() method. So if main() method returns something there is no way of catching that return statement. So always main() method return type is void.

**main**: This is the name of java main() method. It's fixed and when we start a java program, JVM checks for the main method.

**String []args**: Java main method accepts a “single” argument of **type** String array. This is also called as java command line arguments.

## Public access modifier examples

```
1 package com.pack1 ;
2
3 public class ClassA
4 {
5     public void meth1()
6     {
7         System.out.println("Class A method Called");
8     }
9     public static void main(String[] args)
10    {
11        ClassA aobj=new ClassA();
12        aobj.meth1();
13    }
14 }
15
```

```
1 package com.pack2;
2
3 public class ClassX
4 {
5
6 }
7
8
```

```
<terminated> ClassA [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (22-Feb-2025, 7:44:41 am - 7:44:42 a
Class A method Called
```

Run from class A

```
1 package com.pack1;
2
3 public class ClassA
4 {
5     public void meth1()
6     {
7         System.out.println("Class A method Called");
8     }
9     public static void main(String[] args)
10    {
11        ClassA aobj=new ClassA();
12        aobj.meth1();
13    }
14 }
15
```

```
1 package com.pack2;
2
3 public class ClassX
4 {
5
6 }
7
8
```

```
<terminated> ClassB [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (22-Feb-2025, 7:45:33 am - 7:4
Class A method Called
```

Above run from class B

```
1 package com.pack1;
2
3 public class ClassA
4 {
5     public void meth1()
6     {
7         System.out.println("Class A method Called");
8     }
9     public static void main(String[] args)
10    {
11        ClassA aobj=new ClassA();
12        aobj.meth1();
13    }
14 }
```

```
1 package com.pack2;
2
3 import com.pack1.ClassA;
4
5 public class ClassX
6 {
7     public static void main(String[] args)
8     {
9         ClassA aobj=new ClassA();
10        aobj.meth1();
11    }
12 }
13
14
```

```
1 package com.pack1;
2
3 public class ClassB
4 {
5     public static void main(String[] args)
6     {
7         ClassA aobj=new ClassA();
8         aobj.meth1();
9     }
10 }
11
```

Console X  
<terminated> ClassX [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (22-Feb-2025, Class A method Called

Above run from class X

So, if we are declaring a method as public we can access it throughout the project.

## Private access modifier examples

```
1 package com.pack1;
2
3 public class ClassA
4 {
5     private void meth1()
6     {
7         System.out.println("Class A method Called");
8     }
9     public static void main(String[] args)
10    {
11        ClassA aobj=new ClassA();
12        aobj.meth1();
13    }
14 }
```

```
1 package com.pack2;
2
3 import com.pack1.ClassA;
4
5 public class ClassX
6 {
7     public static void main(String[] args)
8     {
9         ClassA aobj=new ClassA();
10        aobj.meth1();
11    }
12 }
13
14
```

```
1 package com.pack1;
2
3 public class ClassB
4 {
5     public static void main(String[] args)
6     {
7         ClassA aobj=new ClassA();
8         aobj.meth1();
9     }
10 }
11
```

Console X  
<terminated> ClassA [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (22-Feb-2025, Class A method Called

Run above from class A

```
1 package com.pack1 ;
2
3 public class ClassA
4 {
5     private void meth1()
6     {
7         System.out.println("Class A method Called");
8     }
9     public static void main(String[] args)
10    {
11        ClassA aobj=new ClassA();
12        aobj.meth1();
13    }
14 }
```

```
1 package com.pack2;
2
3 import com.pack1.ClassA;
4
5 public class ClassX
6 {
7     public static void main(String[] args)
8     {
9         ClassA aobj=new ClassA();
10        aobj.meth1();
11    }
12 }
13
14
```

```
1 package com.pack1;
2
3 public class ClassB
4 {
5     public static void main(String[] args)
6     {
7         ClassA aobj=new ClassA();
8         aobj.meth1();
9     }
10 }
11
```

Console X  
<terminated> ClassA [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (22-Feb-2025, 7:48:35 am - 7:4  
Class A method Called

We cannot call a private method in another class which is present in same package

Because of this in above class B we are getting an error.

```
1 package com.pack1 ;
2
3 public class ClassA
4 {
5     private void meth1()
6     {
7         System.out.println("Class A method Called");
8     }
9     public static void main(String[] args)
10    {
11        ClassA aobj=new ClassA();
12        aobj.meth1();
13    }
14 }
```

```
1 package com.pack2;
2
3 import com.pack1.ClassA;
4
5 public class ClassX
6 {
7     public static void main(String[] args)
8     {
9         ClassA aobj=new ClassA();
10        aobj.meth1();
11    }
12 }
13
14
```

```
1 package com.pack1;
2
3 public class ClassB
4 {
5     public static void main(String[] args)
6     {
7         ClassA aobj=new ClassA();
8         aobj.meth1();
9     }
10 }
11
```

Console X  
<terminated> ClassA [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (22-Feb-2025, 7:48:35 am - 7:4  
Class A method Called

We cannot call a private method in another class which is present in another package

Whenever we are declaring a method access modifier as private, we can access only with in same class.

## Default access modifier example

```
1 package com.pack1 ;
2
3 public class ClassA
4 {
5     void meth1(){}
6     public static void main(String[] args)
7     {
8         ClassA aobj=new ClassA();
9         aobj.meth1();
10    }
11 }

1 package com.pack2;
2
3 import com.pack1.ClassA;
4
5 public class ClassX
6 {
7     public static void main(String[] args)
8     {
9         ClassA aobj=new ClassA();
10        aobj.meth1();
11    }
12 }

1 package com.pack1;
2
3 public class ClassB
4 {
5     public static void main(String[] args)
6     {
7         ClassA aobj=new ClassA();
8         aobj.meth1();
9     }
10 }
11 }
```

Console Output:

```
<terminated> ClassA [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe {
Class A method Called
```

Run above in class A



```
1 package com.pack1 ;
2
3 public class ClassA
4 {
5     void meth1(){}
6     public static void main(String[] args)
7     {
8         ClassA aobj=new ClassA();
9         aobj.meth1();
10    }
11 }
12
13
14
15
16
17
```

```
1 package com.pack2;
2
3 import com.pack1.ClassA;
4
5 public class ClassX
6 {
7     public static void main(String[] args)
8     {
9         ClassA aobj=new ClassA();
10        aobj.meth1();
11    }
12 }
13
14
```

ClassB.java X

```
1 package com.pack1;
2
3 public class ClassB
4 {
5     public static void main(String[] args)
6     {
7         ClassA aobj=new ClassA();
8         aobj.meth1();
9     }
10 }
11
```

Console X

```
<terminated> ClassB [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe
Class A method Called
```

Run above in class B

```
1 package com.pack1 ;
2
3 public class ClassA
4 {
5     void meth1(){}
6     public static void main(String[] args)
7     {
8         ClassA aobj=new ClassA();
9         aobj.meth1();
10    }
11 }
12
13
14
15
16
17
```

```
1 package com.pack2;
2
3 import com.pack1.ClassA;
4
5 public class ClassX
6 {
7     public static void main(String[] args)
8     {
9         ClassA aobj=new ClassA();
10        aobj.meth1();
11    }
12 }
13
14
```

ClassB.java X

```
1 package com.pack1;
2
3 public class ClassB
4 {
5     public static void main(String[] args)
6     {
7         ClassA aobj=new ClassA();
8         aobj.meth1();
9     }
10 }
11
```

Console X

```
<terminated> ClassB [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (22-Feb-2)
Class A method Called
```

Error

We access default access modifier only with in the same package

For protected also there are some restrictions please wait

## Why main method is declared as public?

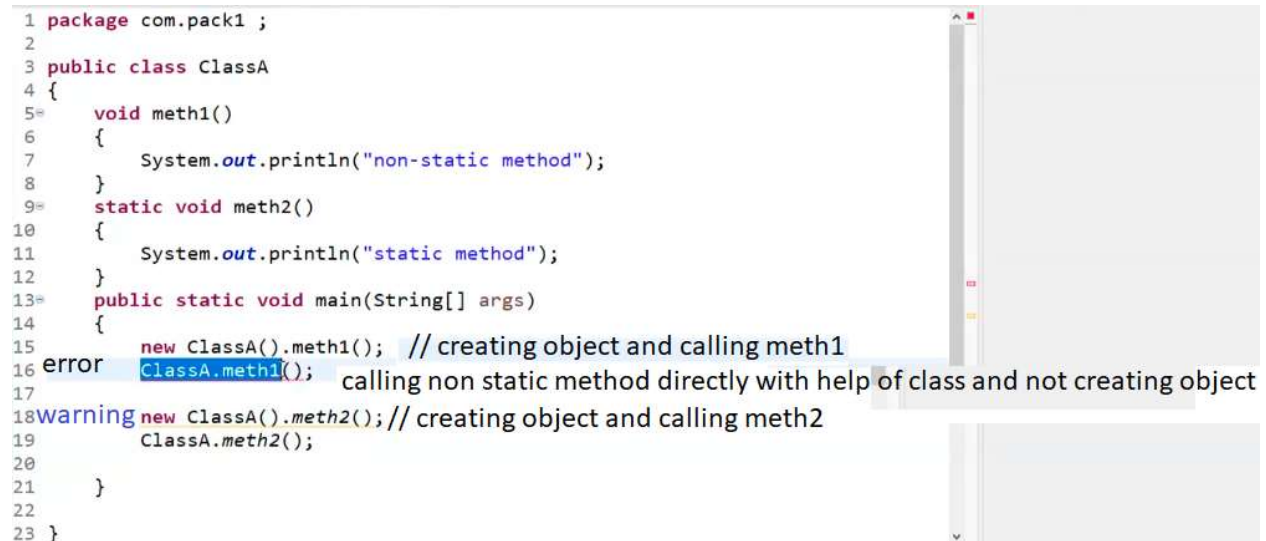
Because it should be available to all the classes which were present in our project

## Why main method is declared as static?

To understand this, we need to have some knowledge of compilation and running.

## Static is a key word

To call a static method, there is no need to create an object. Static methods can be called directly with the help of their respective call names.



```
1 package com.pack1 ;
2
3 public class ClassA
4 {
5     void meth1()
6     {
7         System.out.println("non-static method");
8     }
9     static void meth2()
10    {
11        System.out.println("static method");
12    }
13    public static void main(String[] args)
14    {
15        new ClassA().meth1(); // creating object and calling meth1
16        error ClassA.meth1(); calling non static method directly with help of class and not creating object
17
18        warning new ClassA().meth2(); // creating object and calling meth2
19        ClassA.meth2();
20    }
21 }
22
23 }
```

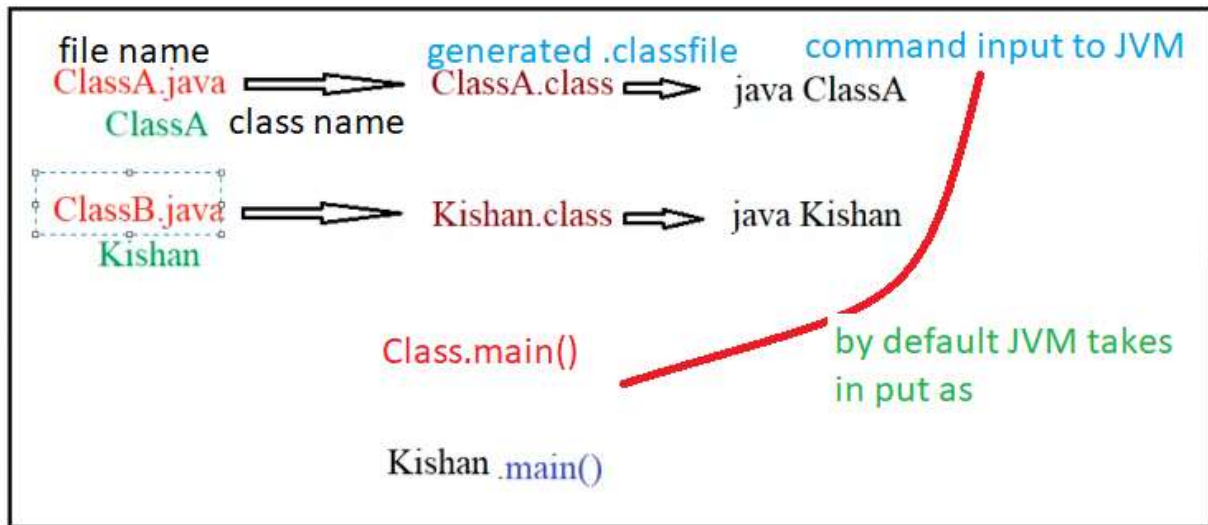
```
1 package com.pack1 ;
2
3 public class ClassA
4 {
5     void meth1()
6     {
7         System.out.println("non-static method");
8     }
9     static void meth2()
10    {
11        System.out.println("static method");
12    }
13    public static void main(String[] args)
14    {
15        new ClassA().meth1();
16
17
18        new ClassA().meth2();
19        ClassA.meth2();
20    }
21 }
22
23 }
```

The top screenshot shows the full code. The bottom screenshot shows the code with the static void meth2() method highlighted. The IDE title bar indicates the application is terminated.

Main is static method, it is a predefined method

If the main method is not declared as static, to call that method, we need an object which will never be created.





## Void

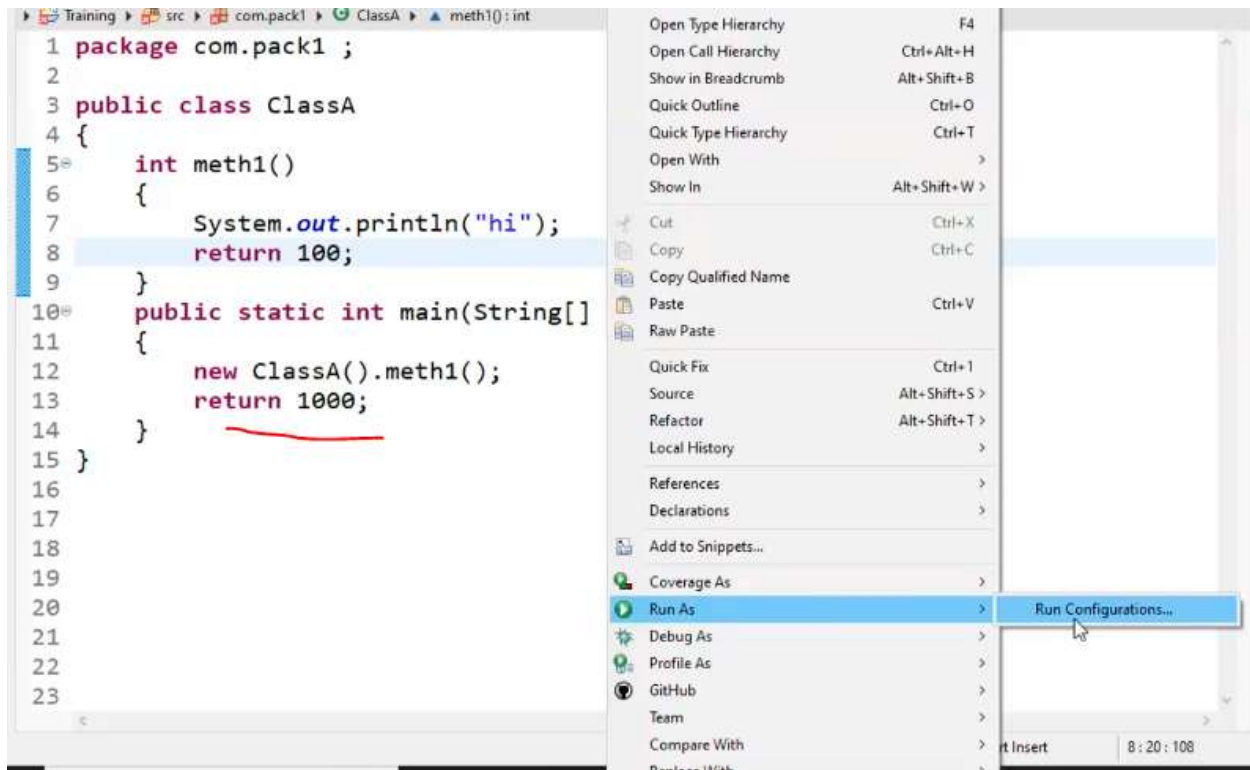
If a method has a return type, it is supposed to return some value, and method is returning it value to its calling method.

If the main method is having any other return type, your program execution will not be ended with main method, your program execution will not be terminated with main method

Your program execution will continue with the main method.

That is not syntactically correct, your main execution should end with the main method

The Void (return type) method does not return anything so the main method should declare as void return type.



Unable to run because of the main method returning some value

## String[] args

If a method is having string array as its parameter, we can pass any kind of data inside the double codes.

## Which of the following main() method syntax are valid?

1. `public static void main(String[] args)`
2. `public static void main(String []args)`
3. `public static void main(String [] args)`
4. `public static void main(String args [])`
5. `public static void main([]String args)`
6. `public static void main(String[] Kishan)`
7. `static public void main(String[] args)`
8. `public static int main(String[] args)`
9. `public final static void main(String[] args)`
10. `Public static void main(String[] args)`
11. `final public static void main(String[] args)`
12. `Final public static void main(String[] args)`
13. `public static void main(String... args)`
14. `public static void mian(String[] args)`
15. `public static void main(String[8] args)`
16. `public static void main(int[] args)`
17. `public static void main()`
18. `public void main(String[] args)`
19. `public static void Main(String[] args)`