

Java main() Method Interview Questions with Answers

Author: [Ramesh Fadatare](#)

Core Java Interview

In this article, we will discuss frequently asked Java *main()* method interview questions with answers for both freshers and experienced.

As we know that Java *main()* method is the entry point of any Java program. Its syntax is always **public static void main(String[] args)**.

YouTube Video

Java main() method interview Questions with Answers

We will discuss below 8 Java main() interview questions and answers:

1. Why the main() method is public static?
2. Can we overload the main() method in Java?
3. Can we declare the main() method as private or protected or with no access modifier?
4. Can we declare the main() method as a non-static?
5. Can we change the return type of the main() method?
6. Can the main() method take an argument other than String array?
7. Can we run define Java Class without the main() method?
8. Can we make the main final in Java?

I will demonstrate all the above questions with examples.

1. Why the main() method is public static?

First, let's see why the main() method is static in Java?

The main() method is static in Java, so the JVM can directly invoke it without instantiating the class's object.

If the main() method is non-static, then JVM needs to create an instance of the class, and there would be ambiguity if the constructor of that class takes an argument – which constructor should

be called by JVM and what parameters should be passed? We know that JVM can't instantiate a Java class without calling a constructor method.

The below example demonstrates why the `main()` method is static in Java?

```
package net.javaguides.corejava;

public class MainMethodDemo {

    public MainMethodDemo(int arg0) {
        //One argument constructor
    }

    public MainMethodDemo(int arg0, int arg1) {
        //Two arguments constructor
    }

    public MainMethodDemo(String arg[]) {
    }

    public void main(String...args) {
        //Non Static main method
    }
}
```

Now, let's see why the `main()` method is public?

We know that anyone can access/invoke a method having *public* access specifier. The `main()` method is public in Java because it has to be invoked by the JVM. So, if `main()` is not public in Java, the JVM won't call it.

2. Can we overload the `main()` method in Java?

Yes, We can overload the *main()* method. A Java class can have any number of *main()* methods. But to run the java class, the class should have a *main()* method with signature as **public static void main(String[] args)**.

The below diagram demonstrates that the *main()* method can be overloaded:

```
MainMethodDemo.java
1 package net.javaguides.corejava;
2
3 public class MainMethodDemo {
4     public static void main(String[] args) {
5         System.out.println("Hello World !");
6     }
7
8     public static void main(int args)
9     {
10         System.out.println("Another main method");
11     }
12
13     public static double main(int i, double d)
14     {
15         System.out.println("Another main method");
16
17         return d;
18     }
19 }
20
```

Problems @ Javadoc Declaration Console

<terminated> MainMethodDemo (1) [Java Application] C:\Program Files\Java\jdk1.8.0_172\bin\javaw.exe (12-Oct-2018, 7:04:00 PM)
Hello World !

Let's see a simple example to demonstrate *main()* can be overloaded:

```
package net.javaguides.corejava;

import java.util.Arrays;

public class MainMethodDemo {

    /** Actual main method with String[] args**/
    public static void main(String[] args) {
        System.out.println("String[] args main method called");
        main(new Double[] {
            1.0,
            2.0,
            3.0
        });
    }

    /** Overloaded main method with Double[] args**/
    public static void main(Double[] args) {
        System.out.println("Double[] args main method called");
        System.out.println(Arrays.toString(args));
    }
}
```

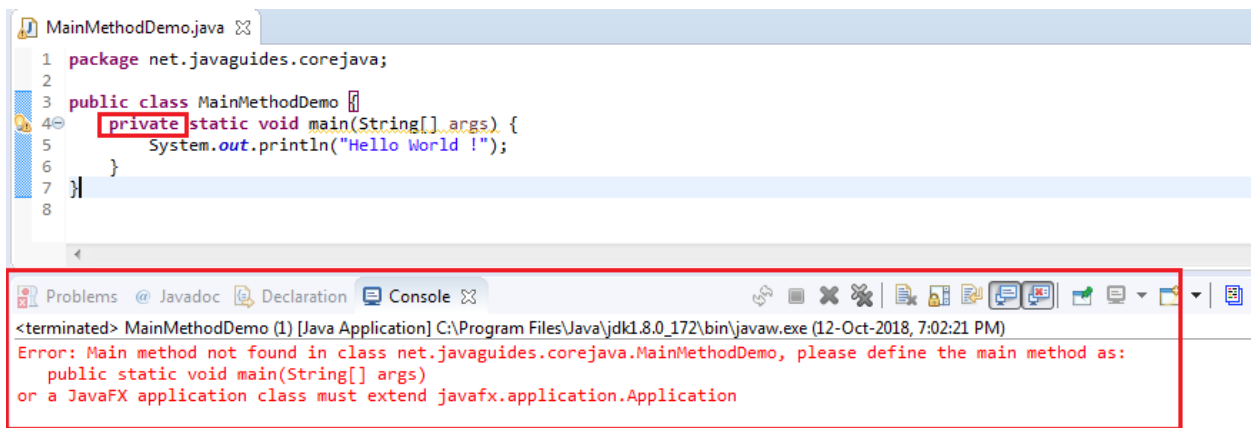
Output:

```
String[] args main method called
Double[] args main method called
[1.0, 2.0, 3.0]
```

3. Can we declare the `main()` method as private or protected or with no access modifier?

No, the `main()` method must be **public**. You can't define the `main()` method as **private** or **protected** or with no access modifier. This is because to make the `main()` method accessible to JVM.

The below diagram shows runtime error, if you define the `main()` method other than public.



```
1 package net.javaguides.corejava;
2
3 public class MainMethodDemo {
4     private static void main(String[] args) {
5         System.out.println("Hello World !");
6     }
7 }
8
```

Problems Javadoc Declaration Console

<terminated> MainMethodDemo (1) [Java Application] C:\Program Files\Java\jdk1.8.0_172\bin\javaw.exe (12-Oct-2018, 7:02:21 PM)

Error: Main method not found in class net.javaguides.corejava.MainMethodDemo, please define the main method as:
public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application

4. Can we declare the `main()` method as a non-static?

No, the `main()` method must be declared as **static** so that JVM can call the `main()` method without instantiating its class. If you remove 'static' from the `main()` method signature, the compilation will be successful but the program fails at runtime.

The below diagram demonstrates that the `main()` method should be static otherwise JVM will throw runtime error:

```
1 package net.javaguides.corejava;
2
3 public class MainMethodDemo {
4     public void main(String[] args) {
5     }
6 }
7
8
```

main() method must be declared as static so that JVM can call main() method without instantiating it's class

<terminated> MainMethodDemo [Java Application] C:\Program Files\Java\jdk1.8.0_172\bin\javaw.exe (12-Oct-2018, 6:56:34 PM)
Error: Main method is not static in class com.javaguides.jdbc.batch.MainMethodDemo, please define the main method as:
public static void main(String[] args)

5. Can we change the return type of the main() method?

No, the return type of the *main()* method must be **void** only. Any other type is not acceptable.

The below diagram demonstrates that the *main()* method should have a **void** return type:

```
1 package net.javaguides.corejava;
2
3 public class MainMethodDemo {
4     public static int main(String[] args) {
5         System.out.println("Hello World !");
6         return 0;
7     }
8 }
9
```

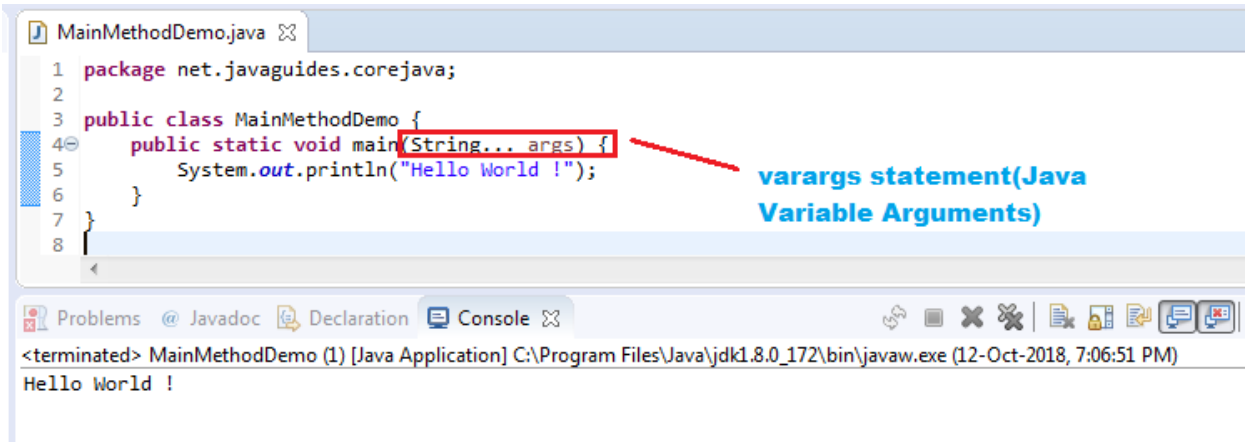
when return type is int

<terminated> MainMethodDemo (1) [Java Application] C:\Program Files\Java\jdk1.8.0_172\bin\javaw.exe (12-Oct-2018, 7:05:15 PM)
Error: Main method must return a value of type void in class net.javaguides.corejava.MainMethodDemo, please define the main method as:
public static void main(String[] args)

6. Can the main() method take an argument other than String array?

No, an argument of the *main()* method must be a String array. But, from the introduction of **var args**, you can pass **var args** of string type as an argument to the *main()* method. Again, **var args** are nothing but the arrays.

The below diagram demonstrates that the *main()* method should have an argument as String array or var args:



```
1 package net.javaguides.corejava;
2
3 public class MainMethodDemo {
4     public static void main(String... args) {
5         System.out.println("Hello World !");
6     }
7 }
8
```

varargs statement (Java Variable Arguments)

<terminated> MainMethodDemo (1) [Java Application] C:\Program Files\Java\jdk1.8.0_172\bin\javaw.exe (12-Oct-2018, 7:06:51 PM)
Hello World !

7. Can we run define Java Class without the *main()* method?

No, We cannot define a class without the *main()* method starting from Java 7. In the previous release of Java, we can have *Static Initializers* as an alternative:

```
public class MainMethodDemo
{
    static
    {
        System.out.println("Static Initializer");
        System.exit(0);
    }
}
```

Output: (From JDK 7)

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
```

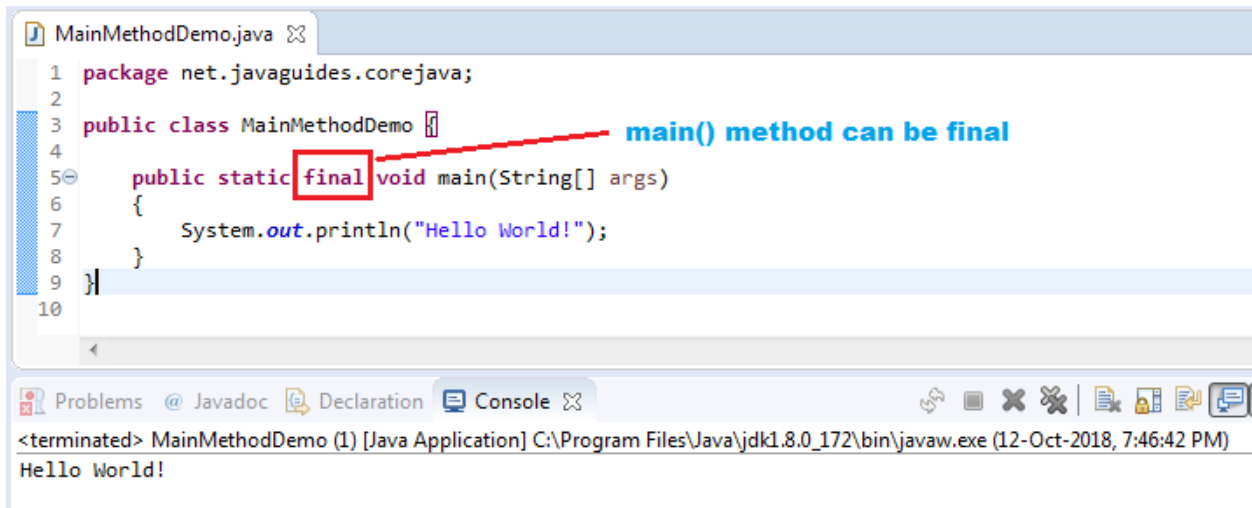
Output: (JDK 6)

Static Initializer

8. Can we make the main final in Java?

Yes, you can make the main() method final.

The below diagram demonstrates that we can have the main() method as final in Java.



```
1 package net.javaguides.corejava;
2
3 public class MainMethodDemo {
4
5     public static final void main(String[] args)
6     {
7         System.out.println("Hello World!");
8     }
9 }
10
```

The screenshot shows an IDE window titled "MainMethodDemo.java". The code defines a package, a public class, and a public static final void main method that prints "Hello World!". A red box highlights the word "final" in the method signature, and a red arrow points from it to the text "main() method can be final". Below the code editor, the "Console" tab is active, showing the output "Hello World!" after the program has terminated.