Compiler Vs Interpreter

A compiler is a tool/software that converts the whole source code at one shot from the high level programming language (like English) to the intermediate language (bytecode in Java).

The successful compilation results in a `.class` file - is called as the bytecode, and that is what given to the Interpretor (`java` executable) - which is nothing but the JRE (Java Runtime Environment).

Demonstrating the Happy Path flow - a Java class with a `main()` method having 3 executable statements

```
raghavan.muthu@Raghavans-MacBook-Pro compilerDemo % pwd
/Users/raghavan.muthu/raghs/study/javapgms/basics/compilerDemo
raghavan.muthu@Raghavans-MacBook-Pro compilerDemo % ls -ltrh
total 8
drwxr-xr-x  4 raghavan.muthu  staff   128B 27 Aug 15:37 example2-
        FileNameWrong
drwxr-xr-x  5 raghavan.muthu  staff   160B 27 Aug 15:45 example1-
        HappyPath
drwxr-xr-x  5 raghavan.muthu  staff   160B 27 Aug 16:00 example3-
        HappyPath-ClassNameNonCompliant
drwxr-xr-x  4 raghavan.muthu  staff   128B 27 Aug 16:03 example4-
        PrivateClass
drwxr-xr-x  7 raghavan.muthu  staff   224B 27 Aug 17:03 example5-
        DefaultClass
drwxr-xr-x  6 raghavan.muthu  staff   192B 27 Aug 17:19 example6-
        ManyClasses
-rw-r--r--  1 raghavan.muthu  staff   228B 27 Aug 17:21
        HelloWorld.java
raghavan.muthu@Raghavans-MacBook-Pro compilerDemo %
raghavan.muthu@Raghavans-MacBook-Pro compilerDemo % bat
        HelloWorld.java
```

```
       │ File: HelloWorld.java
```

```
   1   │ public class HelloWorld
   2   │ {
   3   │     public static void main(String... args)
   4   │     {
   5   │         System.out.println("Hello World!");
   6   │         System.out.println("Statement 2 in the main
       method!");
   7   │         System.out.println("Statement 3 in the main
       method!");
   8   │     }
   9   │ }
  10   │
```

Attempt to compile the Source file

```
raghavan.muthu@Raghavans-MacBook-Pro compilerDemo % javac
        HelloWorld.java
raghavan.muthu@Raghavans-MacBook-Pro compilerDemo %
raghavan.muthu@Raghavans-MacBook-Pro compilerDemo % ls -ltrh *.class
-rw-r--r--  1 raghavan.muthu  staff   524B 27 Aug 17:22
        HelloWorld.class
```

Listing the files in the directory

```
raghavan.muthu@Raghavans-MacBook-Pro compilerDemo % ls -ltrh
total 16
drwxr-xr-x  4 raghavan.muthu  staff   128B 27 Aug 15:37 example2-
        FileNameWrong
drwxr-xr-x  5 raghavan.muthu  staff   160B 27 Aug 15:45 example1-
        HappyPath
```

```
drwxr-xr-x  5 raghavan.muthu  staff   160B 27 Aug 16:00 example3-
        HappyPath-ClassNameNonCompliant
drwxr-xr-x  4 raghavan.muthu  staff   128B 27 Aug 16:03 example4-
        PrivateClass
drwxr-xr-x  7 raghavan.muthu  staff   224B 27 Aug 17:03 example5-
        DefaultClass
drwxr-xr-x  6 raghavan.muthu  staff   192B 27 Aug 17:19 example6-
        ManyClasses
-rw-r--r--  1 raghavan.muthu  staff   228B 27 Aug 17:21
        HelloWorld.java
-rw-r--r--  1 raghavan.muthu  staff   524B 27 Aug 17:22
        HelloWorld.class
raghavan.muthu@Raghavans-MacBook-Pro compilerDemo %
```

Executing the class `HelloWorld`

```
raghavan.muthu@Raghavans-MacBook-Pro compilerDemo % java HelloWorld
Hello World!
Statement 2 in the main method!
Statement 3 in the main method!
raghavan.muthu@Raghavans-MacBook-Pro compilerDemo %
```

> *Note*: Nothing different here. The class was compiled and executed successfully, producing the 3 different output in the console as expected.

Purposefully making some mistakes in the source file `HelloWorld.java`

```
raghavan.muthu@Raghavans-MacBook-Pro compilerDemo % bat
        HelloWorld.java
──────┬──────────────────────────────────────────────────────────────────────────
      │ File: HelloWorld.java
──────┼──────────────────────────────────────────────────────────────────────────
   1  │ public class HelloWorld
   2  │ {
   3  │     public static void main(String... args)
   4  │     {
   5  │         System.out.println("Hello World!");
   6 ~│         System.out.println("Statement 2 in the main method!")
   7 ~│         System.out.println("Statement 3 in the main method!")
   8  │     }
   9  │ }
  10  │
──────┴──────────────────────────────────────────────────────────────────────────
```

```
raghavan.muthu@Raghavans-MacBook-Pro compilerDemo % git diff
        HelloWorld.java
diff --git a/HelloWorld.java b/HelloWorld.java
index 8cadb85..ccb9389 100644
--- a/HelloWorld.java
+++ b/HelloWorld.java
@@ -3,8 +3,8 @@ public class HelloWorld
     public static void main(String... args)
     {
         System.out.println("Hello World!");
-        System.out.println("Statement 2 in the main method!");
-        System.out.println("Statement 3 in the main method!");
+        System.out.println("Statement 2 in the main method!")
+        System.out.println("Statement 3 in the main method!")
     }
 }
```

The above snippets shows the following

- The first one was the actual /raw source code content where you could see the semicolon missing on 2nd and 3rd `System.out.println()` statements - line # 6 and 7.

- The second snippet is the `git diff` that produces the difference in the contents of the file `HelloWorld.java - with the current version Vs the previously committed version, and we can infer the following.
  - The character on the left hand side – indicates the line was deleted. The `#2` and `#3` `System.out.println()` statements with the valid semicolon at the end were deleted in this version,
  - The character + indicates the line being added - the `#2` and `#3` `System.out.println()` *without* semicolon were added in this version.

  Basically it shows the difference of the two lines in such a way that the one with a semicolon was removed. `

Let us remove the `.class` file purposefully to ensure that there is no `.class` file is present in the folder

```
raghavan.muthu@Raghavans-MacBook-Pro compilerDemo % rm -rf *.class
raghavan.muthu@Raghavans-MacBook-Pro compilerDemo % ls -ltrh *.class
zsh: no matches found: *.class
```

Attempt to compile the Source file `HelloWorld.java`

```
raghavan.muthu@Raghavans-MacBook-Pro compilerDemo % javac
        HelloWorld.java
HelloWorld.java:6: error: ';' expected
        System.out.println("Statement 2 in the main method!")
                                                            ^
HelloWorld.java:7: error: ';' expected
        System.out.println("Statement 3 in the main method!")
                                                            ^
2 errors
raghavan.muthu@Raghavans-MacBook-Pro compilerDemo % ls -ltrh
        HelloWorld*
-rw-r--r--  1 raghavan.muthu  staff   226B 27 Aug 17:31
        HelloWorld.java
```

> *Note*: As expected, the source file did *not* get compiled successfully, and the Compiler threw the errors - all at once. You can see that there were two errors - one for each Statement in the lines #6 and #7 respectively.

As a result, there was no `.class` file generated, which happens only after a class gets successfully compiled. The last command shows that there is only file matching with the pattern `HelloWorld*`, which is the source file and NOT any `.class` files.

A dare attempt to execute the Class - `HelloWorld` :)

```
raghavan.muthu@Raghavans-MacBook-Pro compilerDemo % java HelloWorld
Error: Could not find or load main class HelloWorld
Caused by: java.lang.ClassNotFoundException: HelloWorld
raghavan.muthu@Raghavans-MacBook-Pro compilerDemo %
```

> *Note*: Since there is no compiled version of the class named `HelloWorld`, the JRE is not able to pick up (or load) the .class file, and hence it throws a meaningful error on the console → “Could not find or load main class HelloWorld”.