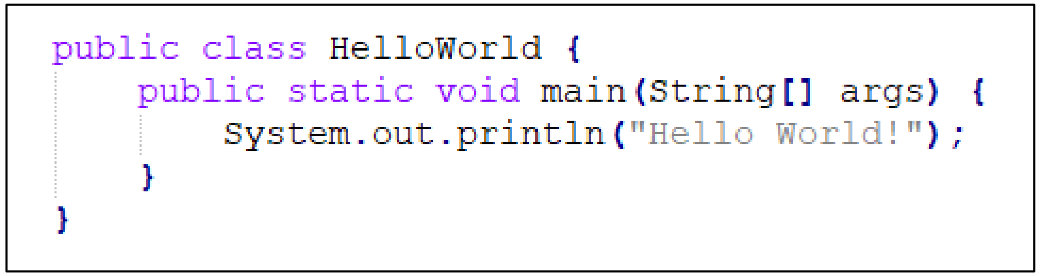
Java 21

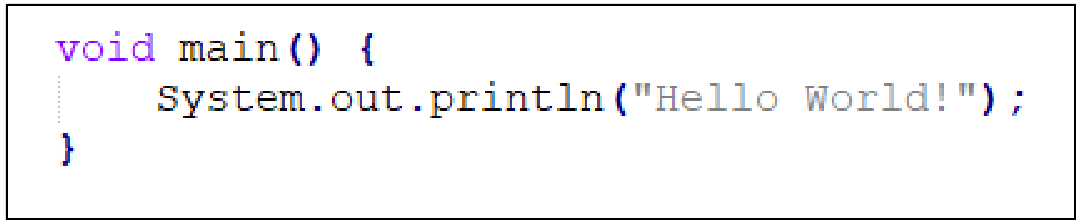
Unnamed Classes and Instance Main Methods

Unnamed Classes and Instance Main Methods

- This is a **preview** feature.
  - https://openjdk.org/jeps/445



```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```



```java
void main() {
    System.out.println("Hello World!");
}
```

Unnamed Classes and Instance Main Methods

- Java supports both "programming in the small" (variables, methods, control flow etc..) and "programming in the large" (classes, interfaces, packages, modules etc..).

- The goal is to focus on the "programming in the small" by reducing ceremony/scaffolding for those learning the language.

- Constructs such as classes, access modifiers such as *public* and keywords such as *static* relate to "programming in the large" and should only be encountered when required.

Unnamed Classes and Instance Main Methods

- Java supports both “programming in the small” (variables, methods, control flow etc..) and “programming in the large” (classes, interfaces, packages, modules etc..).

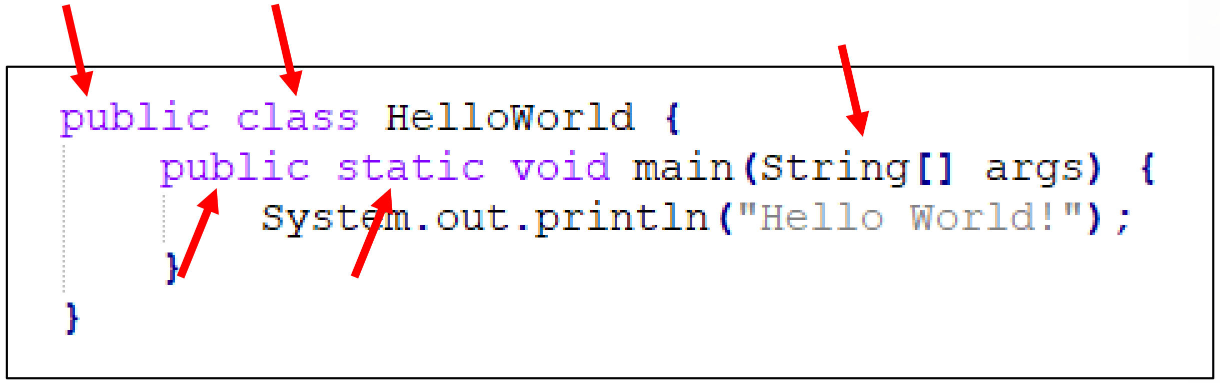
- The goal is to focus on the “programming in the small” by reducing ceremony/scaffolding for those learning the language.

- Constructs such as classes, access modifiers such as *public* and keywords such as *static* relate to “programming in the large” and should only be encountered when required.

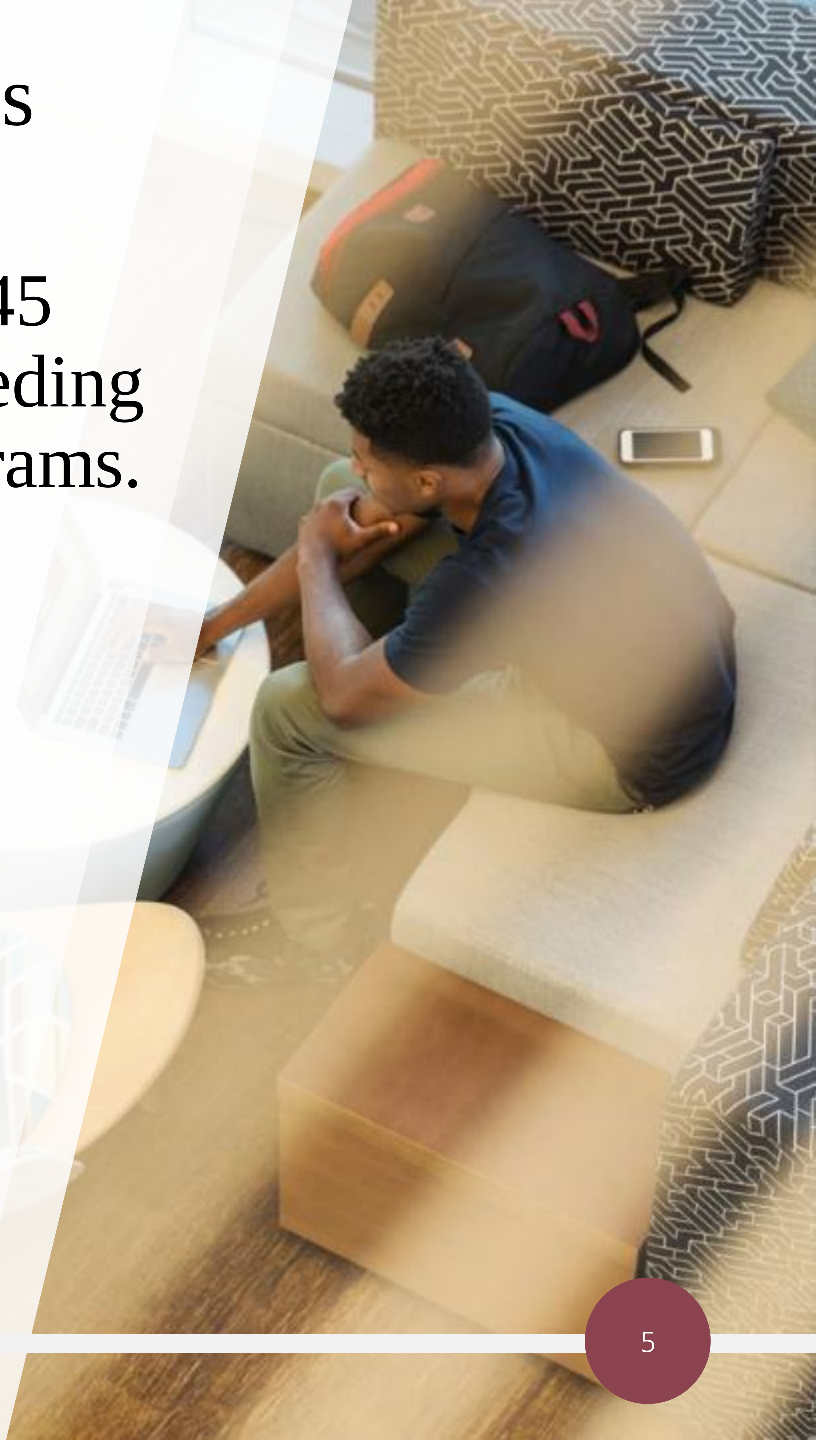
Unnamed Classes and Instance Main Methods

- In effect, make Java easier to learn. To this end, JEP 445 enables learners to write their first programs without needing to understand language features designed for large programs.

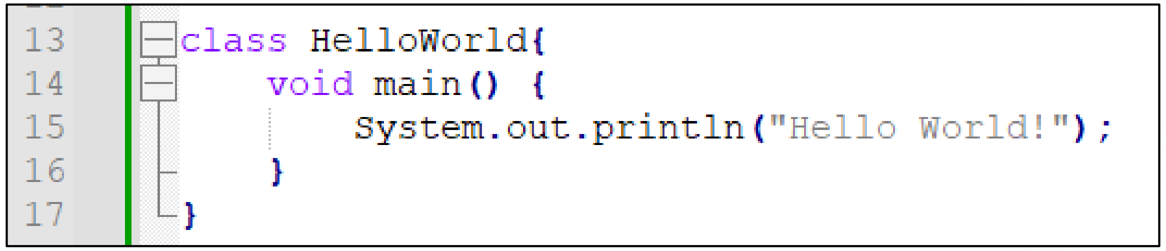
- Basic programs in a concise manner.



```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```



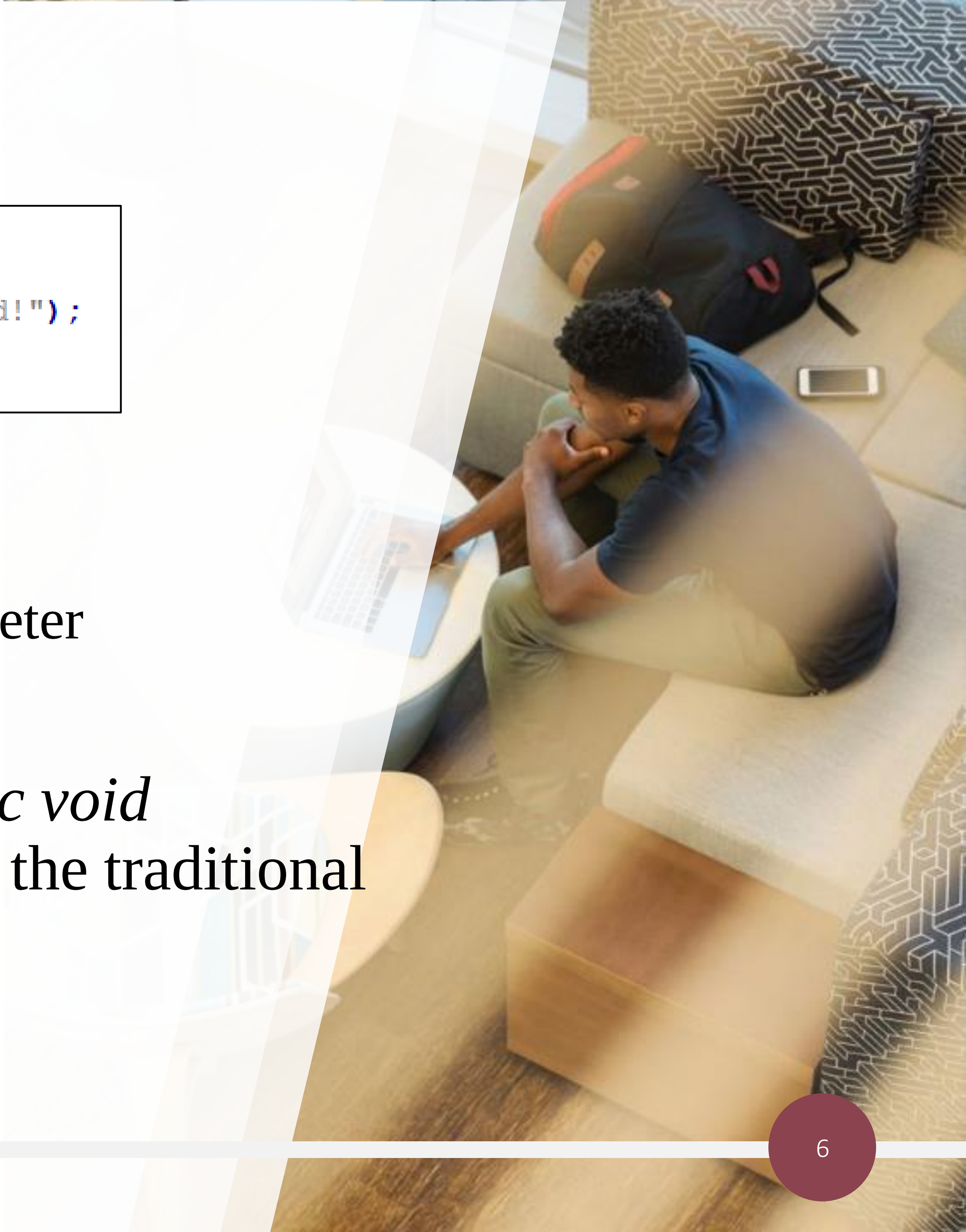
Instance Main Methods



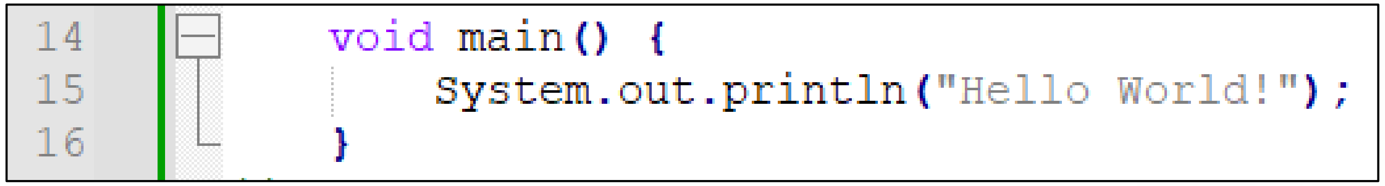
```
class HelloWorld{
    void main() {
        System.out.println("Hello World!");
    }
}
```

- Instance main methods:
  - no need for *static*, *public* or a *String* [] parameter

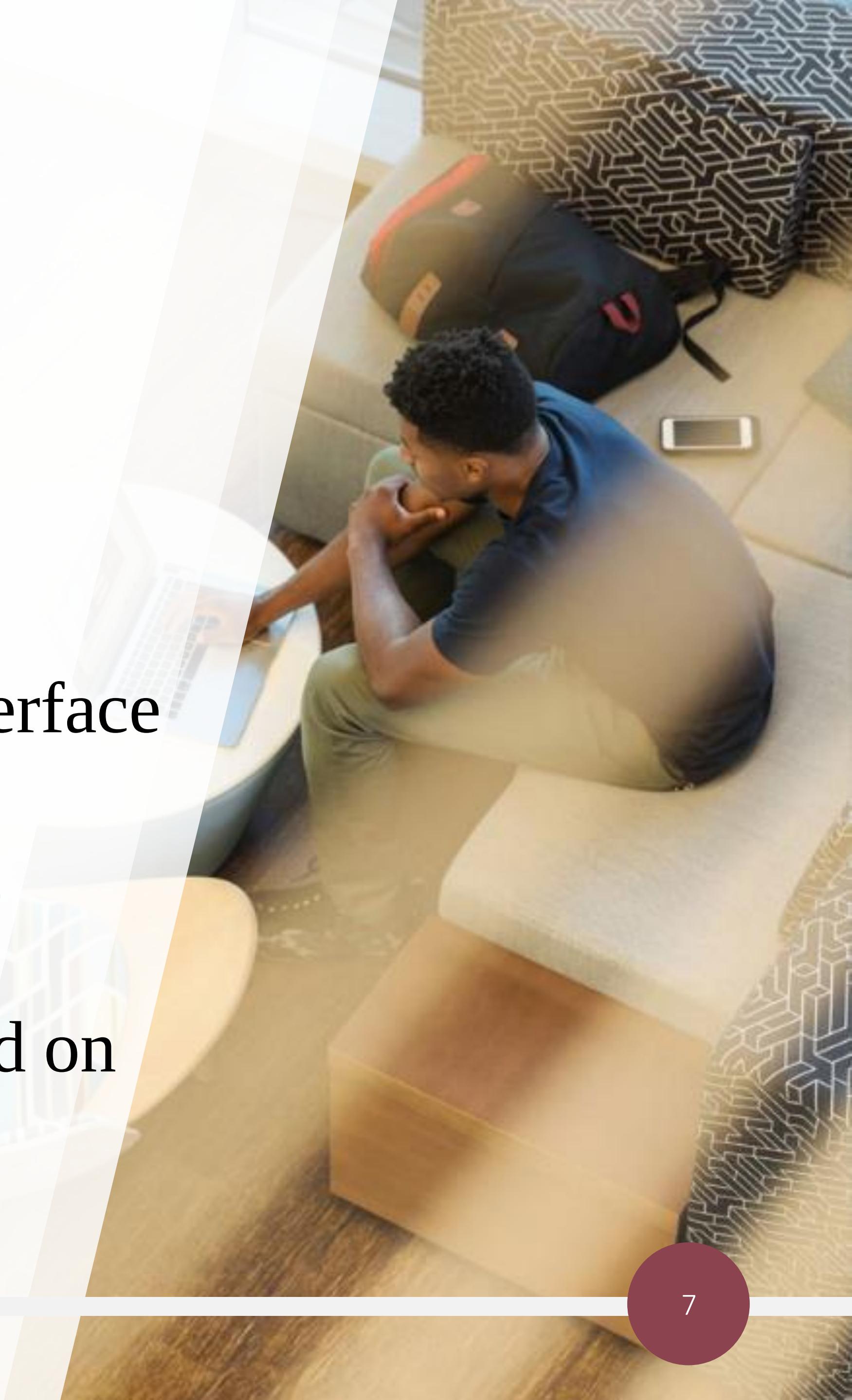
- If you have both the traditional *public static void main(String[] args)* and the instance *main()*, the traditional version takes precedence.



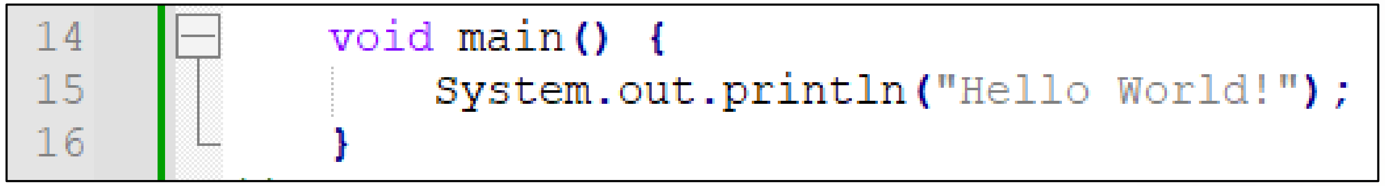
Unnamed Classes


```
void main() {
    System.out.println("Hello World!");
}
```

- Unnamed classes:
  - extend from *Object* and cannot implement an interface
  - are *final* and reside in the unnamed package
  - their *.class* name on the hard disk depends on the filename – for example, if the above code is in *HelloWorld.java*, *HelloWorld.class* is created on the hard disk



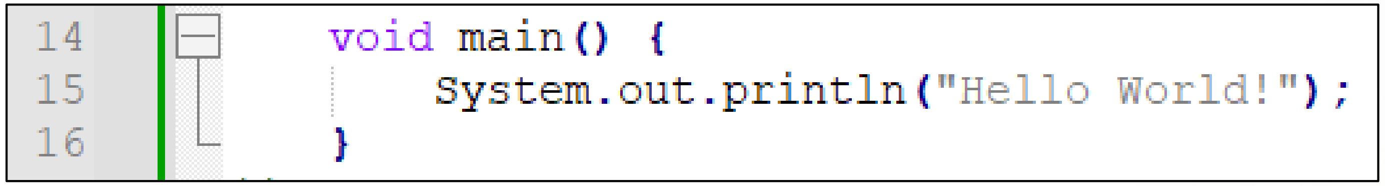
Unnamed Classes



```java
void main() {
    System.out.println("Hello World!");
}
```

- Unnamed classes:
  - are exactly like normal classes **<u>except</u>** that an unnamed class has only one constructor – the default no-args constructor provided by the compiler.
  - it is an error to explicitly code a constructor, even a no-args constructor.
  - the *this* keyword is still valid.

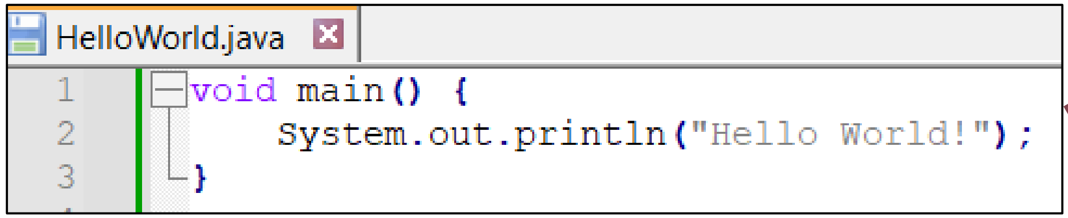
Unnamed Classes



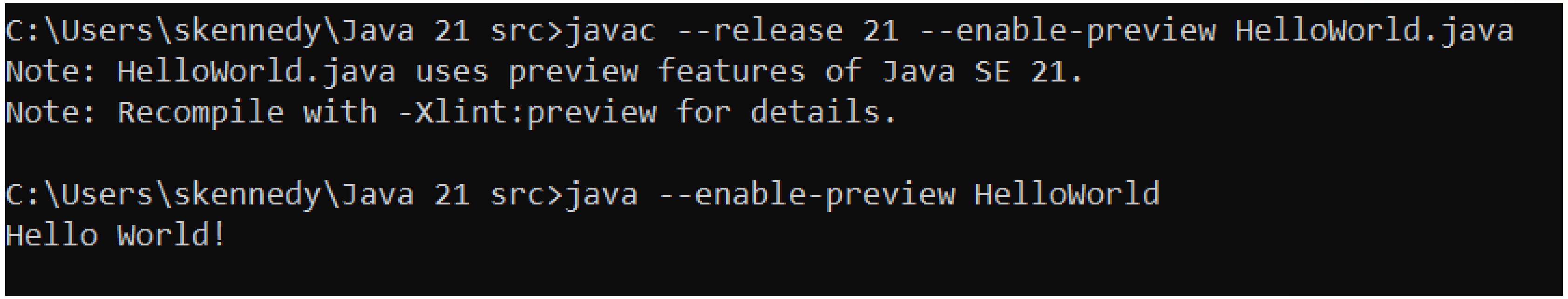
```
void main() {
    System.out.println("Hello World!");
}
```

- Unnamed classes:
  - as code cannot refer to an unnamed class by name, instances of an unnamed class cannot be constructed directly.
  - therefore, such classes are useful for standalone programs or as an entry-point to a program.
  - as a result, unnamed classes must have a *main()* method.

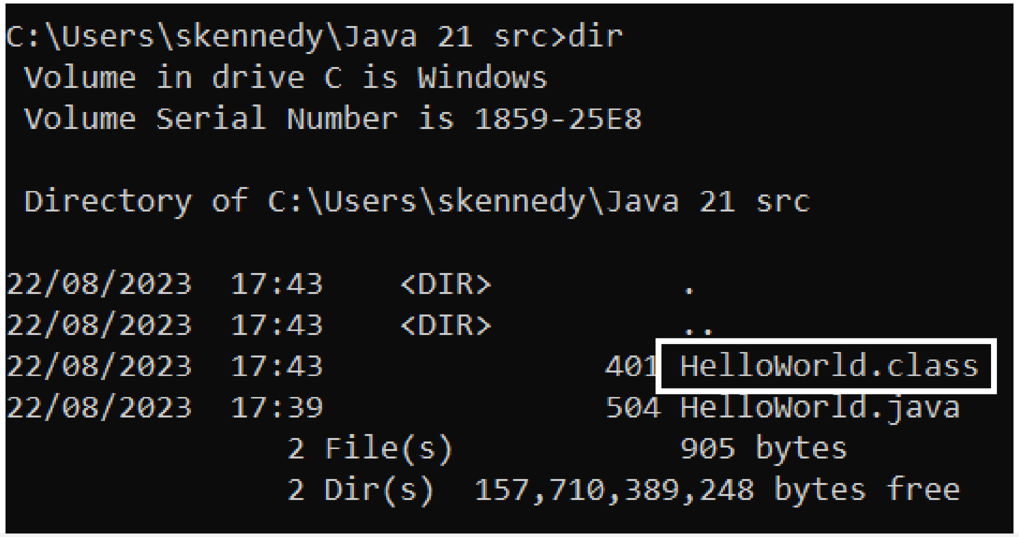
Unnamed Classes



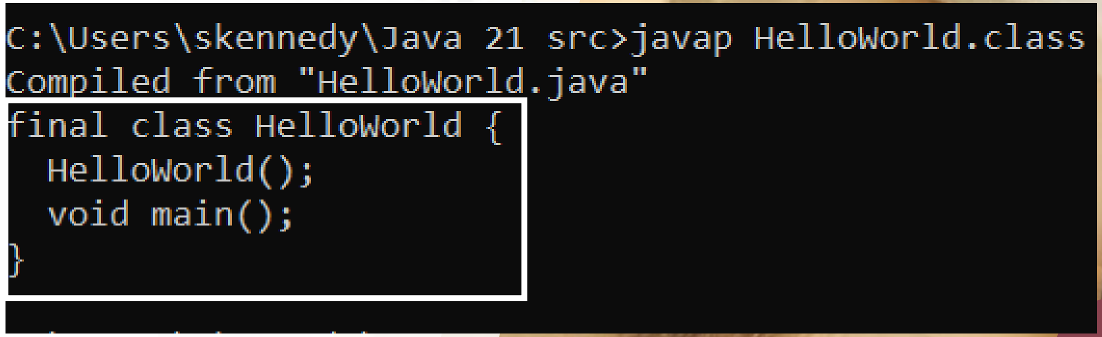
```java
void main() {
    System.out.println("Hello World!");
}
```



```
C:\Users\skennedy\Java 21 src>javac --release 21 --enable-preview HelloWorld.java
Note: HelloWorld.java uses preview features of Java SE 21.
Note: Recompile with -Xlint:preview for details.

C:\Users\skennedy\Java 21 src>java --enable-preview HelloWorld
Hello World!
```



```
C:\Users\skennedy\Java 21 src>dir
 Volume in drive C is Windows
 Volume Serial Number is 1859-25E8

 Directory of C:\Users\skennedy\Java 21 src

22/08/2023  17:43    <DIR>          .
22/08/2023  17:43    <DIR>          ..
22/08/2023  17:43               401 HelloWorld.class
22/08/2023  17:39               504 HelloWorld.java
               2 File(s)            905 bytes
               2 Dir(s)  157,710,389,248 bytes free
```



```
C:\Users\skennedy\Java 21 src>javap HelloWorld.class
Compiled from "HelloWorld.java"
final class HelloWorld {
  HelloWorld();
  void main();
}
```