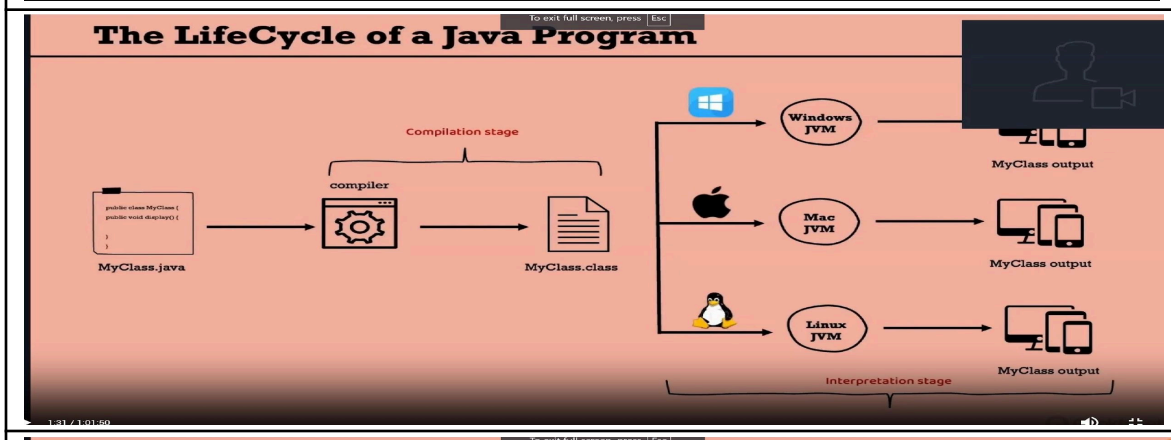
ELUKPO-CHIMA CONSTANCE
Java Report

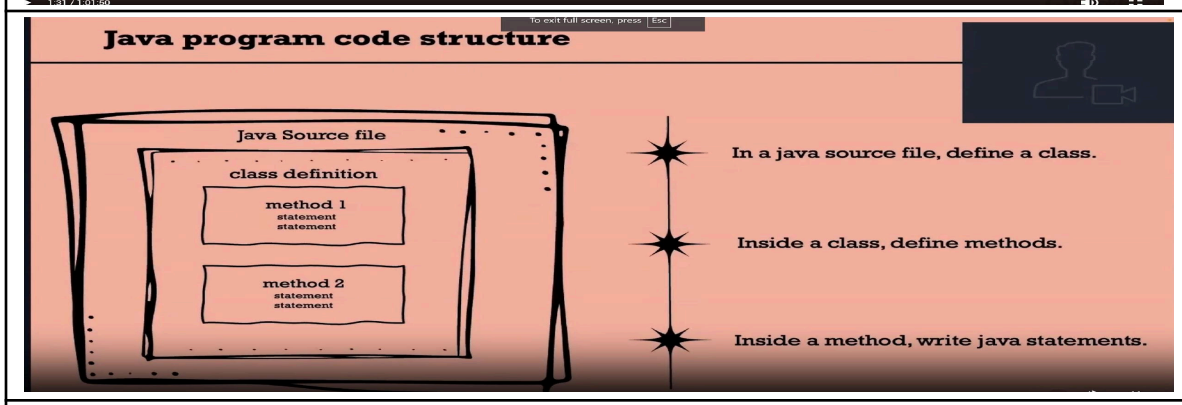
```
PS C:\Users\ENVY\Java> touch HelloWorld.java
Touch Version 5.0 Copyright (c) 1995-2010 Embarcadero Technologies, Inc.

PS C:\Users\ENVY\Java> javac HelloWorld.java;
PS C:\Users\ENVY\Java> javac HelloWorld.java;
PS C:\Users\ENVY\Java> java HelloWorld
Hello World
PS C:\Users\ENVY\Java>
```

Java Installation and Environment Setup

To start programming in Java you need to install the Java Development Kit(JDK). I installe JDK 24 and added it to the system's environment variables (specifically the PATH) so that I can run Java commands from any terminal.
Used this code to verify its installation.



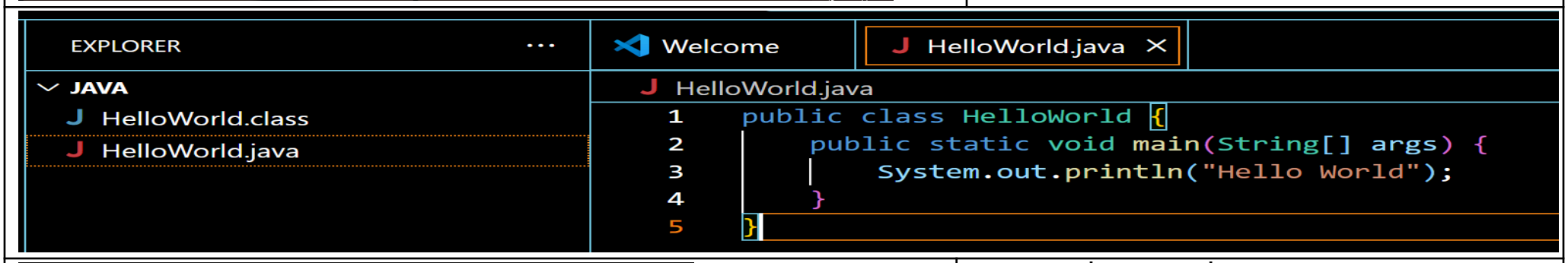
The Java program **life cycle** includes the following components:
Writing Code: The developer writes the .java source file using a text editor or IDE.
Compiling: The javac compiler converts the .java file to bytecode stored in a .class file.
Execution: The **Java Virtual Machine (JVM)** interprets the bytecode and executes the program.
JRE (Java Runtime Environment): Provides the necessary libraries and resources for running Java programs.
JDK (Java Development Kit): Includes the JRE plus development tools like the compiler and debugger.



Each Java file has a class with the same name as the file. The main method is the entry point of any Java application.

A Java program starts with a class.
The main method is the entry point.
Use of semicolons, curly braces, and naming conventions.

Open the terminal or command prompt in the folder where your .java file is, then run: **javac HelloWorld.java**
The javac command compiles the file. If there are no errors, it creates a file: **HelloWorld.class**
Which has been created in the screenshot below. This .class file contains **Java bytecode**, which is the format understood by the **JVM (Java Virtual Machine)**.




```
PS C:\Users\ENVY> jshell
|  Welcome to JShell -- Version 24
|  For an introduction type: /help intro

jshell> /list

jshell> System.out.print("Constance")
Constance
jshell> int myFirstNumber = 5;
myFirstNumber ==> 5

jshell> System.out.print(myFirstNumber);
5

jshell> myFirstNumber = 10;
myFirstNumber ==> 10

jshell> System.out.print(myFirstNumber);
10
jshell> myFirstNumber = 1000;
myFirstNumber ==> 1000

jshell> System.out.print(myFirstNumber);
1000
jshell>  /list
|  Error:
|  illegal start of expression
|   /list
|   ^

jshell> /list

   1 : System.out.print("Constance")
   2 : int myFirstNumber = 5;
   3 : System.out.print(myFirstNumber);
   4 : myFirstNumber = 10;
   5 : System.out.print(myFirstNumber);
   6 : myFirstNumber = 1000;
   7 : System.out.print(myFirstNumber);

jshell> int myFirstNumber = 1000;
myFirstNumber ==> 1000
```

Integer Declaration and Reassignment

int is used to store whole numbers.

I reassigned the same variable multiple times to show how variables can hold new values.

```
jshell> myFirstNumber = 10 + 5;
myFirstNumber ==> 15

jshell> myFirstNumber = (10 + 5) + (2 * 10);
myFirstNumber ==> 35
```

Arithmetic Operations

Demonstrates how Java handles order of operations (BODMAS).
Shows that expressions can be evaluated and assigned to variables.

```
jshell> int mySecondNumber = 12;
mySecondNumber ==> 12

jshell> int myThirdNumber = 6;
myThirdNumber ==> 6

jshell> /var
|    int myFirstNumber = 35
|    int mySecondNumber = 12
|    int myThirdNumber = 6


jshell> int myTotal = myFirstNumber + mySecondNumber + myThirdNumber;
myTotal ==> 53
```

Multiple Integer Variables

Declared multiple variables and used them in an arithmetic expression.
Useful in understanding how values can be combined.

```
jshell> int myMinIntValue = Integer.MIN_VALUE;
myMinIntValue ==> -2147483648

jshell> int myMaxIntValue = Integer.MAX_VALUE;
myMaxIntValue ==> 2147483647

jshell> System.out.print ("Integer Minimum Value = " + myMinIntValue);
Integer Minimum Value = -2147483648
jshell> System.out.print ("Integer Minimum Value = " + Integer.MIN_VALUE);
Integer Minimum Value = -2147483648
jshell> System.out.print("Integer Value Range(" + Integer.MIN_VALUE + "to" + Integer.MAX_VALUE + ")" );
Integer Value Range(-2147483648to2147483647)
jshell> System.out.print ("Busted Max Value = " + (myMaxIntValue + 1));
Busted Max Value = -2147483648
jshell> System.out.print ("Busted Min Value = " + (myMinIntValue - 1));
Busted Min Value = 2147483647
jshell> System.out.print ("Integer Maximum Value = " + Integer.MAX_VALUE);
Integer Maximum Value = 2147483647
```

Integer Limits

int myMinIntValue = Integer.MIN_VALUE; int myMaxIntValue = Integer.MAX_VALUE;
Every primitive type in Java has a min and max limit.
These lines show the lowest and highest values an int can hold.
System.out.print ("Busted Max Value = " + (myMaxIntValue + 1));
Demonstrates **overflow** (value exceeds the max limit and wraps around).

```
jshell>  byte myMinByteValue = Byte.MIN_VALUE, myMaxByteValue = Byte.MAX_VALUE;
myMinByteValue ==> -128
myMaxByteValue ==> 127

jshell> short firstShort = 1; int firstInteger = 2;
firstShort ==> 1
firstInteger ==> 2
```

Byte and Short

byte and short are smaller versions of int.
Useful when optimizing memory usage.

```
jshell> byte byteValue = 10;
byteValue ==> 10

jshell> short shortValue = 20;
shortValue ==> 20

jshell> int intValue = 30;
intValue ==> 30

jshell> long longTotal = 50000L + 10L * (byteValue + shortValue + intValue);
longTotal ==> 50600

jshell>  int sumofThree = byteValue + shortValue + intValue;
sumofThree ==> 60

jshell> longTotal = 50000L + (10 * sumofThree);
longTotal ==> 50600
```

Long Type and Arithmetic

long is used for very large integers.
The L suffix specifies it's a long literal.

```
jshell> System.out.print("Float Value Range(" + Float.MIN_VALUE + " to " + Float.MAX_VALUE + ")");
Float Value Range(1.4E-45 to 3.4028235E38)
jshell>  int myIntValue = 5; float myFloatValue = 5; double myDoubleValue =5;
myIntValue ==> 5
myFloatValue ==> 5.0
myDoubleValue ==> 5.0

jshell> myFloatValue = 5f
myFloatValue ==> 5.0

jshell> myDoubleValue = 5d
myDoubleValue ==> 5.0

jshell>  float myOtherFloatValue = (float)5.25;
myOtherFloatValue ==> 5.25

jshell> int myIntValue = 5; float myFloatValue = 5f; double myDoubleValue =5d;
myIntValue ==> 5
myFloatValue ==> 5.0
myDoubleValue ==> 5.0
```

Float and Double Types

float and double are for decimal values.
Use f and d to declare float and double literals explicitly.
float myOtherFloatValue = (float)5.25;
Casting was used to fix a type conversion error from double to float.

```
jshell> myIntValue = 5 / 2;
myIntValue ==> 2

jshell> myFloatValue = 5f / 2f;
myFloatValue ==> 2.5

jshell> myDoubleValue = 5d / 2d;
myDoubleValue ==> 2.5

jshell> myIntValue = 5 / 3;
myIntValue ==> 1

jshell> myFloatValue = 5f / 3f;
myFloatValue ==> 1.6666666

jshell> myDoubleValue = 5d / 3d;
myDoubleValue ==> 1.6666666666666667
```

Division and Precision

myIntValue = 5 / 2; // Result: 2 (integer division) myFloatValue = 5f / 2f; // Result: 2.5 myDoubleValue = 5d / 2d; // Result: 2.5
Shows the difference in precision between int, float, and double.

```
jshell> double numberofPounds = 200d;
numberofPounds ==> 200.0

jshell> double convertedKilograms = numberofPounds * 0.45359237d;
convertedKilograms ==> 90.718474

jshell> System.out.print("Converted Kilograms = " + convertedKilograms);
Converted Kilograms = 90.718474
jshell>
```

Real-world Calculation

double numberofPounds = 200d; double convertedKilograms = numberofPounds * 0.45359237d;
Converts pounds to kilograms.
Demonstrates practical use of floating-point arithmetic

```
jshell> double anotherNumber= 3_000_000.4_567_890d;
anotherNumber ==> 3000000.456789
```

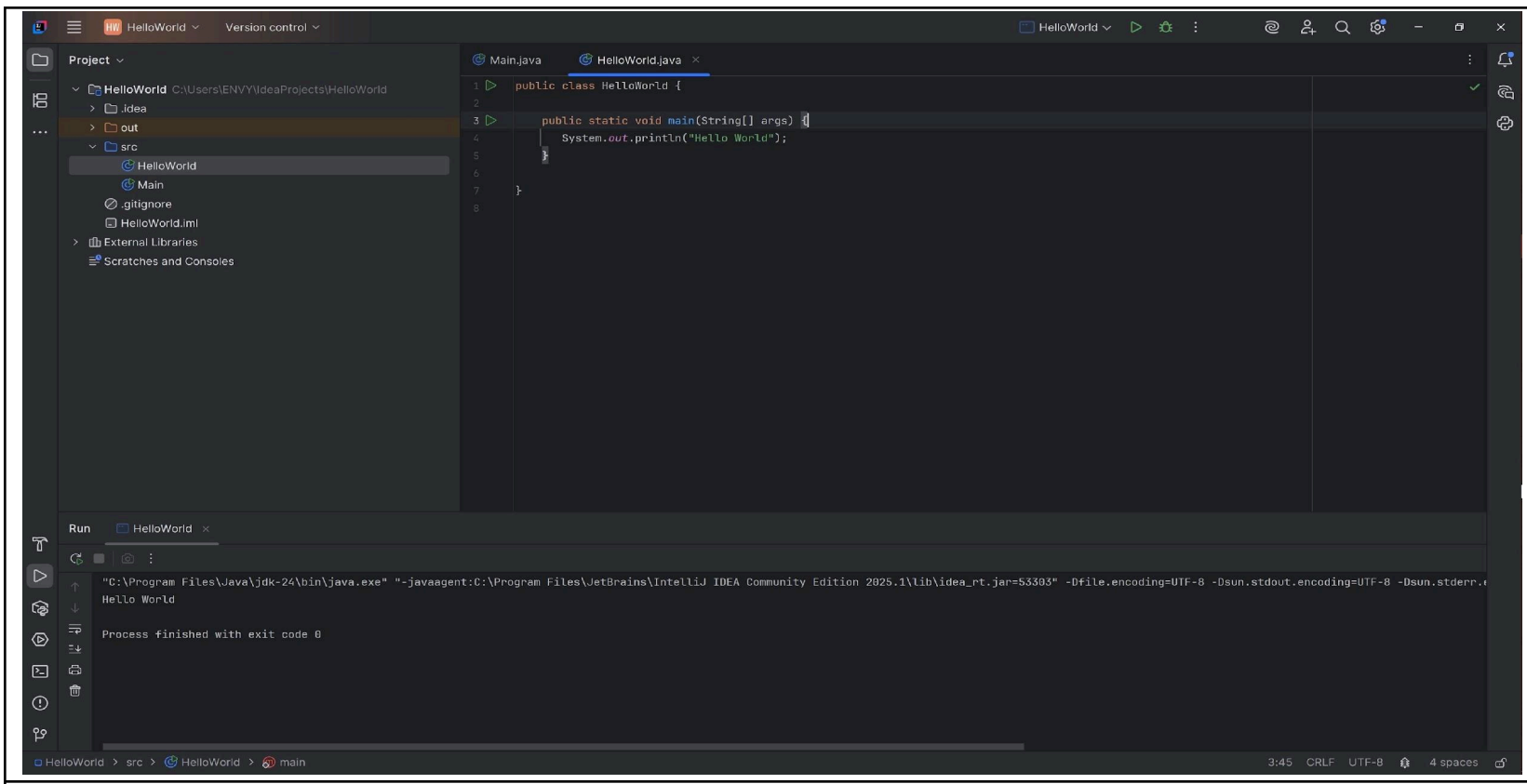
Underscore in Numeric Literals

Java allows underscores in numbers for better readability.

```
jshell> char mychar = 'D';
mychar ==> 'D'

jshell> char myUnicode = '\u0044'
myUnicode ==> 'D'

jshell>
```

Character and Unicode

char myChar = 'D';
char stores a single character.



After installing the Intelliji ide, i did the same ran the same code we ran earlier on, Visual Studio code , on this ide
Created a new project “HelloWorld.java” and typed in the codes. Right Clicked on the src folder and
Manually created the java class file, unlike before where we used the cmd or powershell and the javac
Command to create the java .class file.
In Summary:
Action------ --------- ----------Tool ---------- ----------Result
Created class via GUI -- IntelliJ (right-click src) -- Main.java created
Compiled manually --- IntelliJ Terminal (javac) -- Main.class created
Ran manually --- -- --- IntelliJ Terminal (java) -Output seen: “Hey I’m now running…”