03: Our first Java Program

Important Terms

Hello Java!

Important Terms

Import Things to Know about Java

- Java is an **Object-Oriented Programming** Language.
- An **object** is a block of memory that contains variables and methods.
  - A **method** is an action that an object is allowed to perform.
  - If an object doesn't have a method, then it can't do that action.
- A **class** defines the variables and methods for **objects**.

Import Things to Know about Java

- Every Java program contains at least one class.
- There is exactly one a class containing a **method** called **main**.
- The **main** method is the start of the program.
  - If you aren't sure where a program begins, look for **main**.
- The name of the **class** containing **main** must be spelled identical to the name of the file. This includes capitalization.
- The extension for this file must always be ".java".

Hello Java!

Start Up NetBeans

1. Click the button in NetBeans for a new project.
    - It looks like a brown box with a green plus sign. You'll find it in the menu bar.
2. Make sure that you select "Java" under "Categories".
3. Make sure that you create a "Java Application" under "Projects". Click "Next".
4. Name your project "HelloJava".
    - Make sure that you spell it identical to how it appears in quotes.
    - "H" and "J" are capitalized.
    - There is no space between "Hello" and "Java".
5. Click "Finish".

Raise your hand if you have trouble with any of these steps.

Begin Programming:

Look for this line:

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

Raise your hand if you have trouble finding this line.

Remove a line of code.

Under the line of code that contains **main**, look for the next line.

```
public static void main(String[] args) {
    // TODO code application logic here
```

- Keep the line that contains **main**. Delete the entire line of code that begins with “// TODO ...”
- Make sure that you keep all of the “}”’s after the line containing TODO.

Your code should look like this.

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

}
```

If your code looks like this, click the green arrow on the menu button. This is the “Play” button. If we ever tell you to empty a method, it should look something like this.

Program Output

This is **main** method is empty. The entire program does nothing. When you click “Play”, you should see this:

```
run:
BUILD SUCCESSFUL (total time: 0 seconds)
```

If you do not see “BUILD SUCCESSFUL”, please raise your hand.

Add an output line

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

Write the line between the two curly brackets. The two curly brackets represent the beginning and end of **main**. Don't forget to put a semicolon at the end of your line. Run your program with the green arrow button.

After running.

You should see this. Now you are programming in Java!

```
run:
Hello, world!
BUILD SUCCESSFUL (total time: 0 seconds)
```

If you don't see your message, please raise your hand.

Add more output lines

```
public static void main(String[] args) {
    System.out.println("Hello, world!");
    System.out.println("My name is [YOUR NAME]");
    System.out.println("My favorite movie is [MOVIE]");
    System.out.println("My favorite song is [SONG]");
}
```

Run your program again. Java will execute lines in order from first line in **main** to the last line. All statements in Java must end with a semicolon.

Circle Area Problem.

Empty the `main` method. It should have zero lines in it. This is the same problem that we looked at earlier. Let's write this in Java.

- Imagine that we have a circle.
- The diameter of a circle is 5.
- The area of a circle is $\pi \times radius^2$.
- What is the area?

Circle Area Problem.

See if you can write this on your own without seeing the solution. Make sure that you put a semicolon on the end of each statement.

- Write a variable for a floating point variable named `diameter` that is equal to 5.
- Write a variable for a floating point variable named `radius` that is equal to $diameter/2$.
- Write a variable for a floating point variable named `pi` that is equal to 3.1415.
- Write a variable for a floating point variable named `area` that is equal to $\pi radius^2$.
- We will help you with the print statement.

Circle Area Problem.

The solution.

```
public static void main(String[] args) {
    double diameter = 5;
    double radius = diameter / 2;
    double pi = 3.1415;
    double area = pi * radius * radius;
    System.out.println("The circle area is "
                           + area + " units squared.");
}
```

Circle Area Problem.

The result.

```
The circle area is 19.634375000000002 units squared.
```

That is not good. We should round these values.

Circle Area Problem.

Change this.

```
System.out.println("The circle area is "
                   + area + " units squared.");
```

To this.

```
System.out.println("The circle area is "
                   + Math.round(area)
                   + " units squared.");
```

Circle Area Problem

The result.

```
The circle area is 20 units squared.
```

Could we get better precision? Yes, but this requires some math. Let's try to get this to two decimal places.

1. Multiply the value we wish to round by 100.
2. Call round.
3. Divide the value by 100.

Circle Area Problem

Modify our line containing `Math.round` to this.

```
System.out.println("The circle area is "
                         + Math.round(area * 100) / 100.0
                         + " units squared.");
```

Why do I have to make the last 100 appear as “100.0”? Why can’t it be “100”?

Circle Area Problem

The result.

```
The circle area is 19.63 units squared.
```

This is a good result. We know that it's more than 19 and less than 20. It took a language quirk to get us there.

Language Quirks

Java has many quirks and learning these quirks is part of learning Java.
In English, to make a noun plural, you add an "s" onto the end.
But if you wrote that, "I saw cows, deers, mouses, wolfs, oxs, and fairys traveling to the volcanos," they might help you in more ways than one.
Languages, both natural and in programming, have quirks.

Fill the rest of the hour.

Create a Java program that displays a rocket ship using ASCII art.

```
     |     | |
    / \    | |
   |--o|===|-|
   |---|   | |
  /     \  | |
 | U     | | |
 | S     |=| |
 | A     | | |
 |_______| |_|
  |@| |@|  | |
___________|_|_
```