

Programming CRASH COURSE

Why Programming?

Computers and other electronic devices are finite state machines. This means that at any given moment they can exist in only one of many possible states. Take for example, a light bulb. A light bulb is a binary finite state machine. Binary means 'two' so a light bulb can exist in only one of two possible states: ON or OFF. At their most basic level, computer processors are built from many circuits that can be either ON or OFF. To control these MILLIONS of switches to create the countless states that a computer can be in we use computer programming languages.



There are many different programming languages. Each is designed to fit a different need and has different abilities and limitations. To control the MetalCow robot we will be using Java. Java is a compiled language that is derived from C++. Can you find Java on the Family Tree:

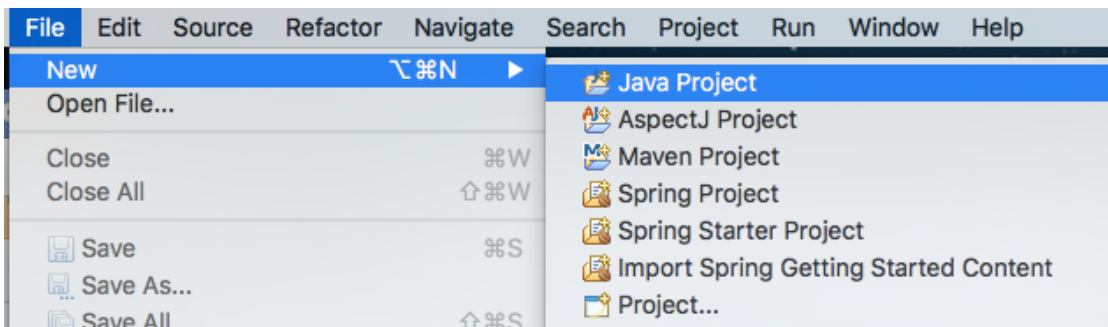
<https://goo.gl/BP0L9M?>

Starting a Project

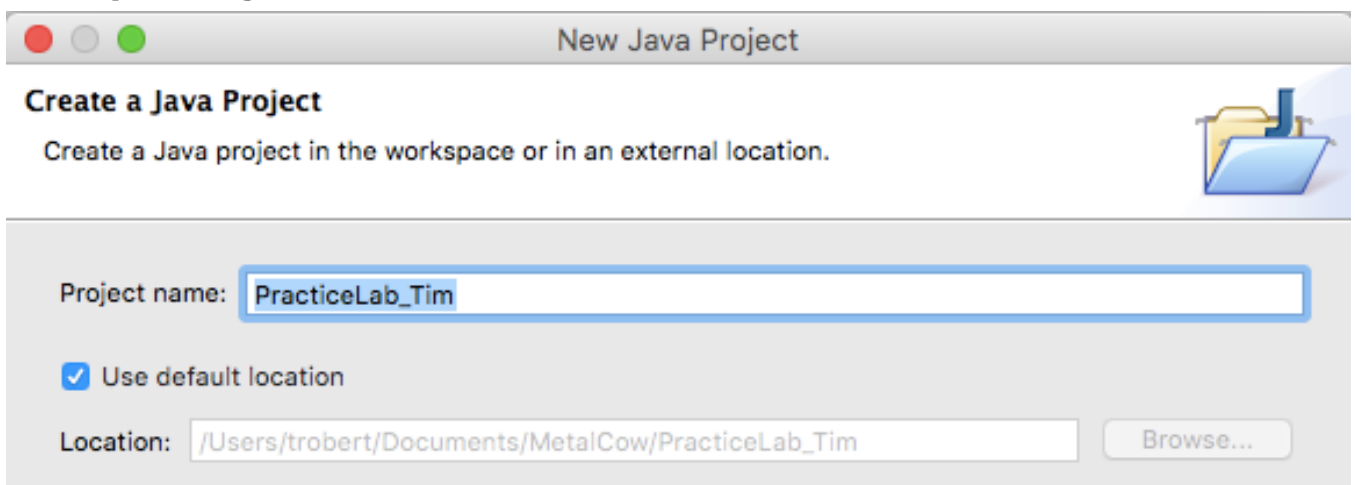
To program in Java all you really need is to have the Java SDK (Software Development Kit) installed on your computer and a text editor such as Notepad. However, just like Photoshop can help us to make images easier than we can make those images in Microsoft Paint, there are software packages called Integrated Development Environments (IDE) that help us to write complex computer programs. For MetalCow we will be using the Eclipse IDE.



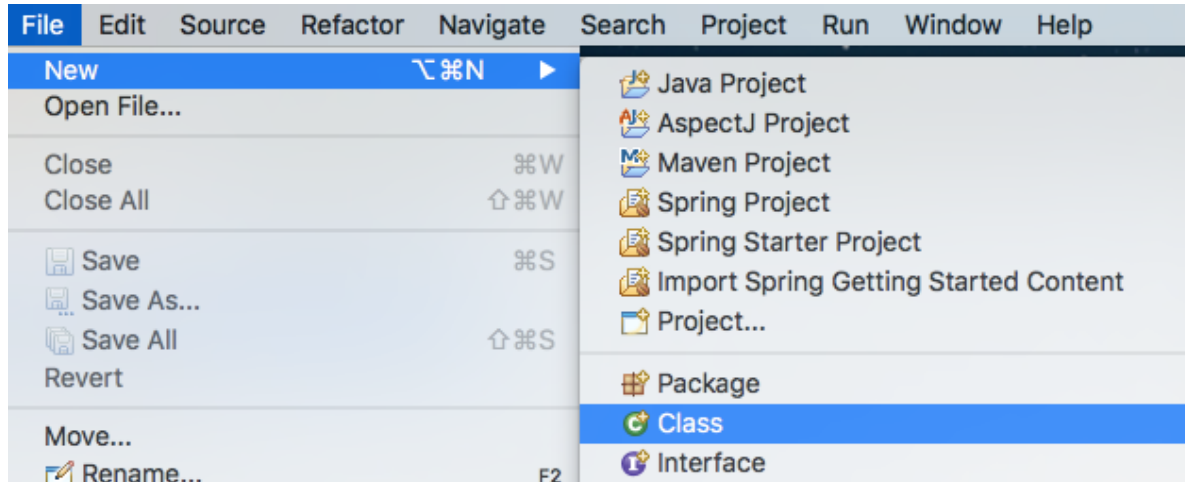
1. Locate the **Eclipse icon and open it**
2. Select **File->New->Java Project**



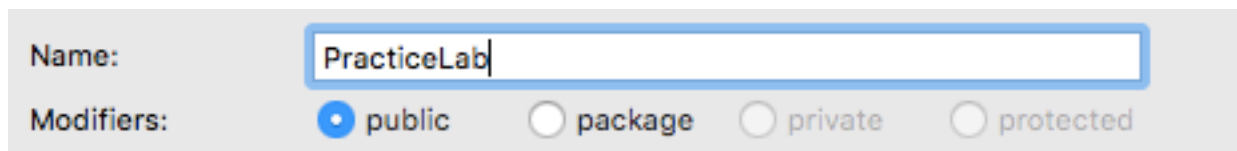
3. Name your Project: **"PracticeLab_*FIRSTNAME*"**



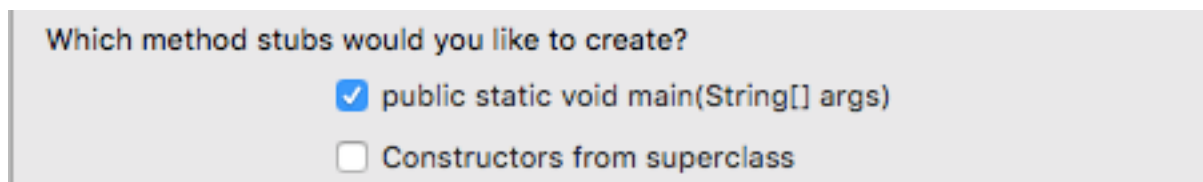
- The remaining options can be left as default. **Click Finish**
- Now on the left side of the screen you should see a new Project listed with your name. Add a new Class file: **Select File->New->Class**



- A dialog will open in the class **Name enter "PracticeLab"**

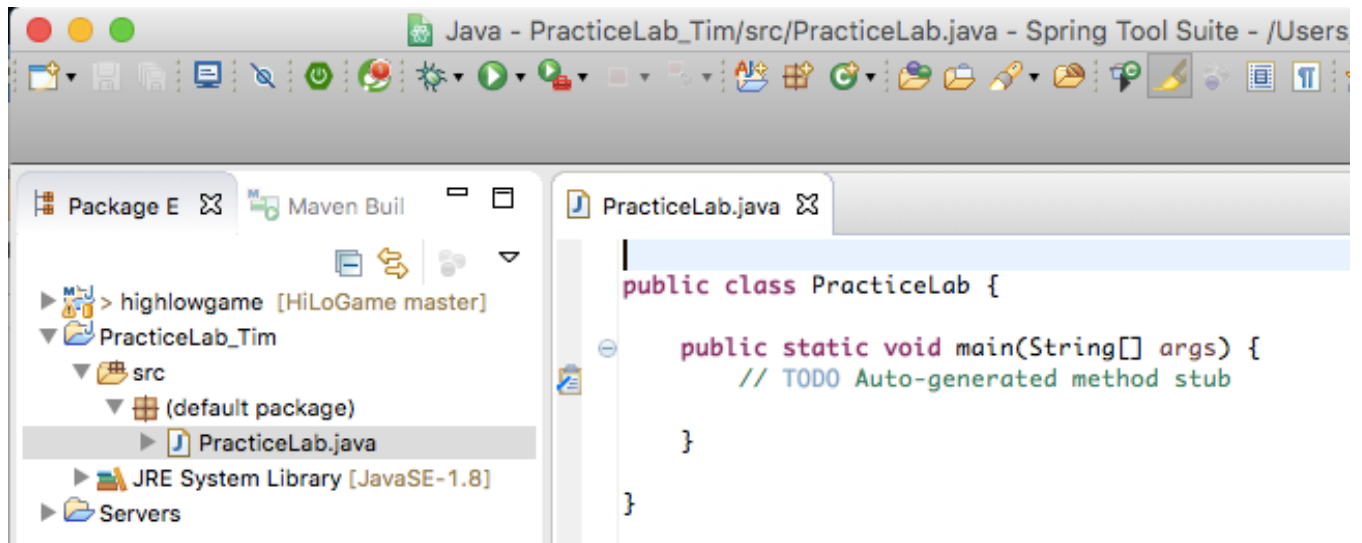


- We will also use the IDE to help us start the file by adding some starter code. To tell the IDE to add the starter code **check the "public static void main(String[] args)" checkbox under "Which method stubs would you like to create?"** as follows:



- Then **click Finish**
- Your project folder on the left should now contain a file called "PracticeLab.java" under the 'src' folder. Also there should be a new tab open called "PracticeLab.java" that contains

some starter code as follows:




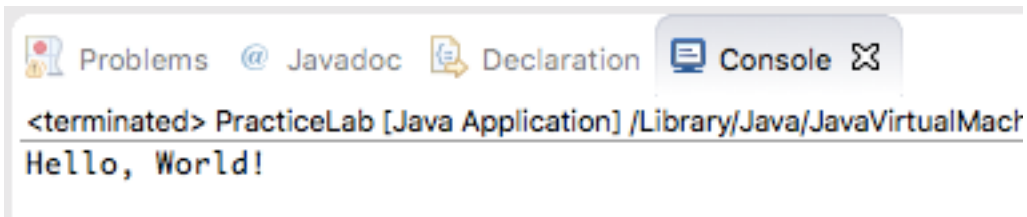
10. You are now ready to write your first program.

Hello, World!


1. It is a tradition in Computer Science that the first program you write when learning a new language is "Hello, World!".
2. In java we can output text to the console (the screen) by using `System.out.print()`; In the project **type the following code** as shown:

```
public class PracticeLab {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        System.out.print("Hello, World!");  
    }  
  
}
```

3. Next let's run our program. **Click the 'Run' command**  and observe the console. If you are prompted to save, click 'ok'. Did you get the following?



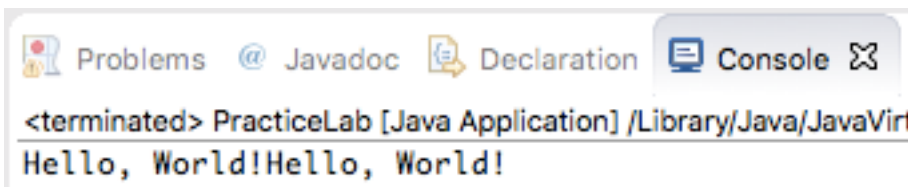
4. While you were typing you may have noticed the following pop-up. This is part of what makes an IDE a powerful tool. The IDE knows many of the operations that `System.out` can perform and lists them so that you can choose the best for what you want to do. Let's see how this might help us.

 *File->Save to save your work. Save often so you don't lose your work if the computer crashes. This can happen and lost work will have to be recreated.*

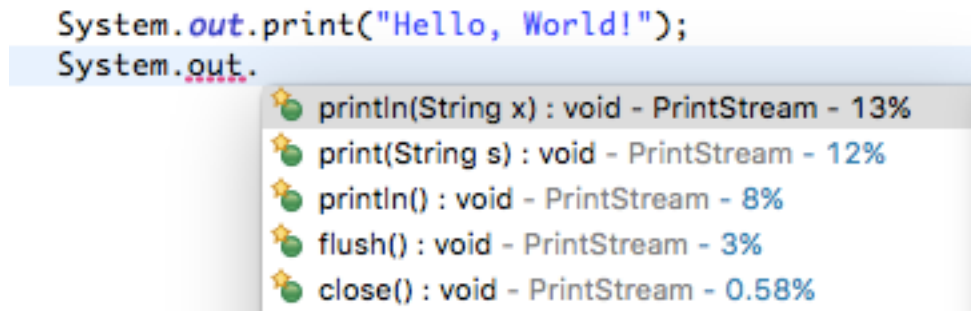
5. **Copy the `System.out.print()`** so that it is in the code twice, as such:

```
public class PracticeLab {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        System.out.print("Hello, World!");  
        System.out.print("Hello, World!");  
    }  
}
```

6. **Run the program and observe the output.** What happened? Why is it printed on the same line twice?



- Let's fix that. In the code, **on a new line type "System.out."** and look at the options in the pop-up:



- What is the difference between `System.out.print(String s)` and `System.out.println(String x)`? **Can you fix the code so that "Hello, World!" prints on different lines?**

Happy Birthday, Ada Lovelace

- Let's continue and build on the lab we have been working on by taking input from the user.
- First let's change "Hello, World!" to "Happy Brithday". **Update the code as shown**, and

```
public class PracticeLab {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        System.out.print("Happy Birthday, |");  
    }  
}
```


run it to make sure it works: }

- Now we will use "String Concatenation" to add on a place holder for the user's name. String Concatenation is adding Strings together. A String is a series of letters and/or numbers. **We**

can use the + sign to do this as shown:

```
public class PracticeLab {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        System.out.print("Happy Birthday, " + "Ada Lovelace");  
    }  
}
```

4. **Does your code work as you expect?** If not ask another student or a mentor to help you.

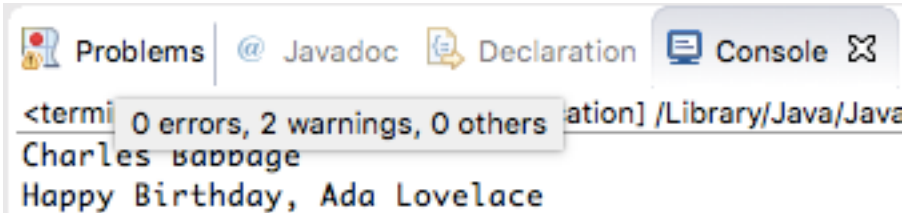
 *Always write code a little at a time and test it as you go. This makes it easy to find and fix mistakes early. Try to fix things yourself first, and then ask for help if you get stuck. Troubleshooting problems is the best way to learn. **An expert is just someone who has made 10,000 mistakes.***

5. Now we will collect the user's name so that the computer can wish them Happy Birthday. To do this we will first need to learn about variables. A variable in computers is just like a variable in Algebra. In the equation: $2+3=Y$
6. Variables in computer programming are names that map to spaces in the computer's memory where we can store values. We will be storing a series of letters, in Java this is called a String, so we will create a new String variable.
7. In the code above the `System.out.print();` **add the following line** `String userName = new String();` **Test your code.** It should run, but the output will not change.
8. Now that you have created the username variable you can store a value in it. **Add a new line** `String userName = "Charles Babbage";` **Test your code.** It should run, but the output will not change.

9. **Add a line as follows:** `System.out.println(userName);` Because `userName` is a `String` we can print it just like we can the string "Ada Lovelace".
10. **Update all of your** `print()` **statements to** `println()` **statements.** Your code should now look as follows:

```
2 public class PracticeLab {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         String userName = new String();
7         userName = "Charles Babbage";
8         System.out.println(userName);
9
10        System.out.println("Happy Birthday, " + "Ada Lovelace");
11    }
12
13 }
```

11. **Test your code.** It should output the following:



The screenshot shows an IDE's Console window with tabs for Problems, Javadoc, Declaration, and Console. The Console tab is active, displaying the output of the Java program. The output consists of two lines: "Charles Babbage" and "Happy Birthday, Ada Lovelace". Above the output, a status bar indicates "0 errors, 2 warnings, 0 others".

```
<termi 0 errors, 2 warnings, 0 others ation] /Library/Java/Java
Charles Babbage
Happy Birthday, Ada Lovelace
```


12. Now, **concatenate** `userName` in place of “Ada Lovelace”, see (line 10):

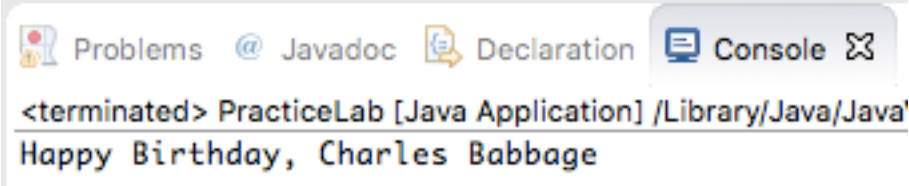
```
1
2 public class PracticeLab {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         String userName = new String();
7         userName = "Charles Babbage";
8         System.out.println(userName);
9
10        System.out.println("Happy Birthday, " + userName);
11    }
12
13 }
```

⚠ Notice, as you type a variable the IDE will highlight other references to that variable in the code. This helps to trace how this variable is used in the program.

13. Next, **remove the test line where we printed** `userName`. You can also **remove the line** `// TODO Auto-generated method stub` The code should look as follows:

```
1
2 public class PracticeLab {
3
4     public static void main(String[] args) {
5         String userName = new String();
6         userName = "Charles Babbage";
7         System.out.println("Happy Birthday, " + userName);
8     }
9
10 }
11
```

14. **Test your code.** The output should match the following:



The screenshot shows the IDE's interface with tabs for Problems, Javadoc, Declaration, and Console. The Console tab is active, displaying the output of the Java application. The output consists of two lines: a terminated message and the actual output of the program.


```
<terminated> PracticeLab [Java Application] /Library/Java/Java'
Happy Birthday, Charles Babbage
```

15. At this point **if your code is not working as expected make sure to call over a mentor.**
-

Happy Birthday to userName!

1. We will now update the code to take user input using the Scanner class.
2. First we will **add a line to ask the user to enter their name:**

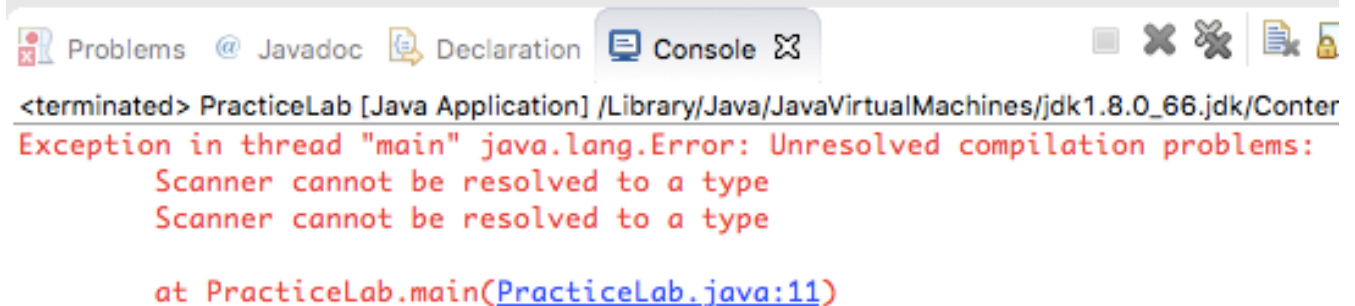
```
1 2
3  public class PracticeLab {
4
5  public static void main(String[] args) {
6      String userName = new String();
7      userName = "Charles Babbage";
8
9      //Ask the user for their name
10     System.out.print("Enter your name: ");
11
12     System.out.println("Happy Birthday, " + userName);
13 }
14
15 }
16
```

 Lines that start with `//` and are green are comments. Comments are little notes that programmers leave in their code to help make it easier to understand. Good code will have useful comments. `//This is a good tip to follow!`

3. Next, we will **create the Scanner to take input from the user:**

```
1 public class PracticeLab {
2
3     public static void main(String[] args) {
4         String userName = new String();
5         userName = "Charles Babbage";
6
7         //Ask the user for their name
8         System.out.print("Enter your name: ");
9
10        //Create a scanner
11        Scanner scanner = new Scanner(System.in);
12
13        System.out.println("Happy Birthday, " + userName);
14    }
15
16 }
```

4. **Notice, that the Scanner on line 11 has a red underline.** If you try to run this code it will not compile, and you will get the following message:



5. To fix this let the IDE help you! **Right-Click on the Scanner error, and choose "Import Scanner".** Scanner is another Class so you need to add a reference to it in your

PracticeLab Class so that Java can find it.

```
//Create a scanner
Scanner scanner = new Scanner(System.in);

//Store the user's input
userName = scanner.next();

System.out.println("Happy Birthday, " + userName);
```

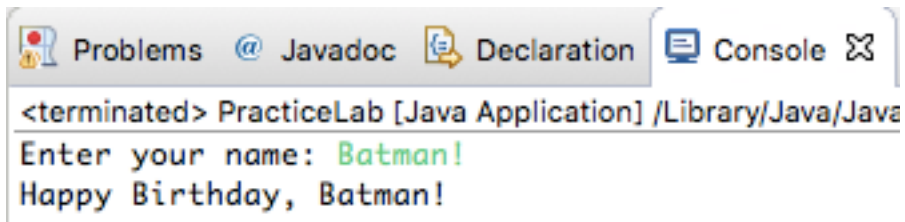
6. Now **your code should look like this**, pay attention to line 1:

```
1 import java.util.Scanner;
2
3 public class PracticeLab {
4
5     public static void main(String[] args) {
6         String userName = new String();
7         userName = "Charles Babbage";
8
9         //Ask the user for their name
10        System.out.print("Enter your name: ");
11
12        //Create a scanner
13        Scanner scanner = new Scanner(System.in);
14
15        System.out.println("Happy Birthday, " + userName);
16    }
17
18 }
```

7. Now that we have a Scanner, use it to **get the user's name from them**:

```
1 import java.util.Scanner;
2
3 public class PracticeLab {
4
5     public static void main(String[] args) {
6         String userName = new String();
7         userName = "Charles Babbage";
8
9         //Ask the user for their name
10        System.out.print("Enter your name: ");
11
12        //Create a scanner
13        Scanner scanner = new Scanner(System.in);
14
15        //Store the user's input userName
16        userName = scanner.next();
17
18        System.out.println("Happy Birthday, " + userName);
19    }
20
21 }
22
```

8. **Test your code.** Type your name in the console and press [Enter]:



The screenshot shows an IDE interface with tabs for Problems, Javadoc, Declaration, and Console. The Console tab is active, displaying the output of the Java application. The text in the console is: <terminated> PracticeLab [Java Application] /Library/Java/Java, followed by the prompt "Enter your name:" and the user input "Batman!". The final output line is "Happy Birthday, Batman!".

9. Congratulations you have now written a computer program that accepts user input!

Continue Learning at Home

If you have access to a computer at school or at home, you can continue to learn and practice there. It is important that students continue learning outside of this session, there are many topics that we will not be able to cover in a 1hour introductory session. Topics such as other data types, math operations, conditional if()...then(), and looping controls. **All of these will be critical to working with the MetalCow robot code.**

The internet is an amazing place to learn programming! ***Don't pay for online classes starting out.*** There are many free online resources which will introduce the core concepts that are needed for MetalCow. If you cannot find Java specifically but your student finds JavaScript or C/C++ classes these are all programming languages in the same family and the core concepts will all transfer. If you are looking for places to learn or have questions feel free to send questions to teammetalcow@gmail.com or on the Metal Cow Slack Channels <http://metalcowrobotics.slack.com>

The following resources should help to get started:

<https://scratch.mit.edu/> - Developed by MIT, Scratch is a programming language with a visual editor specifically designed to teach kids programming concepts and principles at a young age and in a fun way. I have used this at countless STEM education events for kids and adults.

<https://www.codecademy.com/learn/learn-java> - Lessons are free, but you have to create an account. There is no need to upgrade to Pro, just do the main free tutorial lessons.

<http://www.learnjavaonline.org/en/Welcome> - Work the "Learn the Basics" lab activities and feel encouraged to do the advanced topics if you are feeling comfortable.

<https://www.khanacademy.org/computing/computer-programming> - This course uses JavaScript which is a different programming language. However, the programming logic and concepts will be transferable. Also JavaScript is a web language, so if you want to help or play with websites this will be useful.

Weekly Programming Labs and Challenges

Please register for the MetalCow Slack: <http://metalcowrobotics.slack.com>

It is my intention to post activities/labs/challenges in the Slack for those that want to continue learning and preparing for build season during the holidays. I will also post instructions on how to install and setup Eclipse/STS IDEs at home and answer troubleshooting questions there.

These activities will help practice and sharpen the programming concepts that you have learned as well as introduce some of the other sites and topics we will use in MetalCow, such as source code management with <https://github.com/MetalCowRobotics> and the APIs which will be used by the robot.