Java

Intro

My name is John Marchand. The focus of this class will be to learn Java. I have taught this Java Class to many robotics students. We will not focus on robot code in this class; however, I have seen that anyone who understands how to code in Java really well can usually pick up the robot code very quickly.

This class is 2 hours a week; however, you are welcome to send questions in the #programming channel on Slack.

Java Language Reference
https://docs.oracle.com/javase/tutorial/

Student Intro

Name
Grade
Years in Robotics
Programing Experience

HelloWorld

It is convention for your first program to say, “Hello, World!” Every time I learn a new language that is my first program. I have probably done it 50 times in my life. Let’s all get the below program running to make sure our IDE is set up properly.

TODO - Create a file called HelloWorld.java and put the following code in it. Then run it.

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

First, we declare a class called HelloWorld. We will talk about classes later in the course. For now, just accept that needs to match your file name, and Java requires code to be in a class.

Next, we declare function called main. We will also talk about functions later. For now, we will put our code between the { and } after main and the IDE will run our code.

Public and static will also be covered in future classes.

Finally, System.out.println() is a function that will display text on the screen. Please note, Java is case sensitive. Please remember, Java is case sensitive. Also, every statement is terminated with a semicolon.

Variables and Types

Variables are used to store information that your code needs to remember. Most variables are typed. So Strings hold text and Ints hold numbers.

```
        String myName = "John";
        int myAge = 51;

        System.out.println(myName + " is "  + myAge + " years old.");
```

I created two variables and named them. Then, I used them in the print statement with literal text.

TODO – Change this code to be your name and age and then run it.

User Input

Now, lets get some information from the User. Like our println, we will use a predefined class for now and ignore the new statement until later.

```
        Scanner in = new Scanner(System.in);

        System.out.println("What is your name?");
        String name = in.nextLine();

        System.out.println("How Old are you?");
        int age = in.nextInt();

        in.close();
```

TODO – Using the above snippet, can you ask the user their name and age? Then say hi back to them with their name and tell them how old they will be next year. For me it would say “Hi, John! Next year you will be 52.”

TODO – Can you create an adding machine? Ask the user for two numbers, and give them the sum.

Make sure to use in.close() at the end of your main function to avoid resources issues.

Adding Machine

Considering the following code. Two work as expected, one does not. Why?

```
        int sum = firstNumber + secondNumber;
        System.out.println(firstNumber + " + " + secondNumber + " = " + sum);
        System.out.println(firstNumber + " + " + secondNumber + " = " + 
firstNumber + secondNumber);
        System.out.println(firstNumber + " + " + secondNumber + " = " + (
firstNumber + secondNumber));
```

Variables

Variables are used to store values in your program. We think of variables as having a type (String or Int), a Name and a value.

```
        String myName = "John";
        int myAge = 51;
```

Java is a Typesafe language, which means once you declare myName as a string, it cannot store numerical values.

Count to 10

We are going to try several ways to count to 10.

```
        System.out.println(1);
        System.out.println(2);
        System.out.println(3);
        System.out.println(4);
        System.out.println(5);
        System.out.println(6);
        System.out.println(7);
        System.out.println(8);
        System.out.println(9);
        System.out.println(10);
```

Quick and easy, but lets try it with a variable…

```
            int counter = 1;

            System.out.println(counter);
            counter = counter + 1;
            System.out.println(counter);
            counter = counter + 1;
```

These two lines can be copy and pasted as many times as you would like. However, it does result in more lines of code. Lets learn a shortcut. The following two lines are identical

```
        counter = counter + 1;
         counter++;
```

The ++ after a variable increments after the current statement, so we can reduce the previous count to 10 to this.

```
int counter = 1;

System.out.println(counter++);
System.out.println(counter++);
System.out.println(counter++);
```

That's pretty quick and easy, but can we do better? Lets look at loops…

```
int counter = 1;

while ( counter < 11) {
    System.out.println(counter++);
}
```

Or

```
for( int i = 1; i < 11; i++) {
    System.out.println(i);
}
```

Both of these count to 10 quickly for us and are easy to change.

TODO – Write a program to count to 10 using a while loop.

TODO – Write a program to count to 10 using a for loop.

TODO – Write a program to count the odd numbers from 50 to 100 using a while loop.

TODO – Write a program to count to 1000 doubling each time… 1,2,4,8… using a while loop.

Conditions… If

Sometimes you only want to execute code if specific conditions are met. Lets look at some comparison operators.

== equal
!= not equal
< less than
> greater than
<= less than or equal to
>= greater than or equal to

Consider the following if statement.

```
int GullLakeScore = 21;
int LakeViewScore = 3;

if ( GullLakeScore > LakeViewScore )
    System.out.println("Gull Lake Wins!");
System.out.println("Game Over.");
```

What is printed? If you change LakeViewScore to be 31, what is printed?

TODO – Test out this if statement code.

Guessing Game

Consider the following guessing game

```
// this creates a random number from 0 to 9
int answer = (int) (Math.random() * 10 );

int programmersGuess = 7;

if ( programmersGuess == answer )
    System.out.println("You are right.  The answer was " + answer );

if ( programmersGuess != answer )
    System.out.println("You are wrong.  The answer was " + answer );
```

Lets ignore some of the code around creating a random number for now and just accept that it works. This program creates a random number from 0 to 9 and lets the programmer guess at the number and tells us if we were right or wrong.

TODO – Create the Guessing Game and play it a few times.
TODO – Modify the program to tell the user if their guess was too high or too low.
TODO – Change ProgrammersGuess to be user input. Remember to close your input stream.
TODO – Modify the program to allow the user to continue to guess until they get it correct. Let them know if they guess too high or too low.
TODO – Update your game to let the user know how many tries it took them to get the right answer.
BONUS TODO – Update your game to choose a number between 0 and 1000. What is the maximum number of tries it should take you to guess correctly?

Types or Else

We know about Stings and Ints. Now lets add Boolean which holds a value of true or false. We are also adding else to our learning. After an If, sometimes you want to do one thing or the other, else allows you to do that easily.

```
boolean roboticsIsCool = true;
boolean thisClassIsBoring = false;

if ( thisClassIsBoring )
    System.out.println("I want to go home.");
else
    System.out.println("Lets learn Java.");

boolean fiveIsLessThanSix = 5 < 6 ;

if ( fiveIsLessThanSix)
    System.out.println("Duh.");
else
    System.out.println("Houston, we have a problem.");
```

TODO – Without coding, what do you think this will print out?

Floats and doubles allow us to store decimal numbers(14.8) . Ints only store whole numbers (14). We will discuss precision later, but for now use doulbles.

```
double pi = 3.14;
int radius = 5;

// area using doubles
double area = pi*radius*radius;
System.out.println("Area = " + area );

// area using ints
int intPi = (int) pi;
int intArea = intPi*radius*radius;
System.out.println("Int Area = " + intArea );
```

TODO – Write and run this code. Which is more accurate, doubles or ints? What is the value of intPi?

Casting

Java is Typesafe, this means if you try to assign a variable from one type to another it will error. If you consider the code below, the compiler knows you will lose information going from 3.14 to 3. So it throws and error. Casting allows you to say, I really want to do this conversion and I know what I am doing.

```
int intPi = pi;
```

TODO – remove the (int) as above, what error do you get? Does the IDE offer to fix it for you?

Char Type

A Sting is any number of characters put together. There is a type for a single character though. It is called a char.

```
        char myFirstLetter = 'J';
        String message = "My Name is ";
        message = message + myFirstLetter + 'o' + "h" + 'n';
        System.out.println(message);
```

What do you think the output of that is?
Notice strings use “ and chars use ‘. We have some literal characters up there. O, H and N. Even though we don’t put them in a variable, they still have a type. What type is O? H? N?

Arrays

Lets consider the following program to calculate a students average test score.

```
        int test1 = 92;
        int test2 = 88;
        int test3 = 94;

        int testTotal = 0;
        testTotal = testTotal + test1;
        testTotal = testTotal + test2;
        testTotal = testTotal + test3;

        int testAverage = testTotal / 3;

        System.out.println("Average Test Score = " + testAverage);
```

What if I said there were 20 tests? You’d have to create 20 ints each with a unique name. When you are storing many items of the same type, sometimes you want to store the together. To do this we use an array.

```
     int[] tests = new int[3];
```

The type is an array of ints, which we indicate by int[]. We give it a name tests. We then use a new keyword, we’ll discuss later, but we tell Java compiler we want 3 ints. We can now use the following code to store our test scores.

```
        tests[0] = 92;
        tests[1] = 88;
        tests[2] = 94;
```

Arrays start at 0 instead of 1. Now we can calculate the average with this code.

```
        int testTotal = 0;

        for ( int i = 0; i < tests.length; i++ ) {
            testTotal = testTotal + tests[i];
        }

        int testAverage = testTotal / tests.length;
```

One adavantage of using Arrays is that this code will calculate the average regardless of how many test scores are in there.   Tests.length gives us the size of the array, in this case its 3.

You can even set the size of the array at run time.

```
        int numberOfTests = 3;
        int[] tests = new int[numberOfTests];
```

TODO – Write a program that will ask the user how many tests they need to average, then ask them for all the test scores and print the average.

Searching and Sorting

Most programming problems really boil down to either searching or sorting date. Even video games, consider hit boxes, that is the program searching for an area that has a hit.

Consider the following code

```
        String text = "Hello, my name is John";
        char[] textAsArray = text.toCharArray();
```

This creates an array of chars from the string.

TODO – Write a program that will count how many times the letter ‘n’ appears in the sentence.

Tic Tac Toe

Consider the follow char arrays that will allow us to store a tic tac toe board.

```
        char[] topRow = new char[2];
        char[] middleRow = new char[2];
        char[] bottomRow = new char[2];
```

You could assign each char a X, O or Space (‘ ‘); Then you could print out a Tic Tac Toe Board

TODO – Write a program to display a tic tac toe board.
TODO – Update your program to start with a blank board. Then ask the user what square the want to put an X in, then switch to 0.

TODO – Update the program to recognize when someone wins.

2D Arrays and Nested Loops and Debugging

We stored a list of test scores in an array. For a whole class though, you would need an array for each student. Fortunately, you can have 2D arrays.

```
int[][] studentTestScores = new int[5][3];
```

This creates a way to store 3 test scores for 5 students. How do we loop through them all? We use nested loops, consider the following.

```
// create 3 random test scores for each of the 20 students
for ( int i = 0; i < studentTestScores.length; i ++) {
    for ( int j = 0; j< studentTestScores[i].length; j++ ) {
        studentTestScores[i][j] = (int) (80 + ( Math.random() * 20 ));
    }
}
```

This creates 3 random test scores for each student.

Debugging allows you to watch Java execute each line of code. You can set a break point by clicking on the far left of the numbers on screen or press F9 on a line of code. After pressing F5 to start, it will stop on that line. You can use F10 to execute the program one line at a time

TODO – Write the above program and set a break point on the first for statement. Step through and experiment with the debugger.

TODO – Have the program print out the average for each student.

BONUS TODO – Update your tic tac toe program to use char[][] ticTacToeBoard.

Functions

Sometimes we want to group together some code and call it many times. Java like most languages allows us to do this using functions. Consider the following simple function called add

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

        int firstNumber = 7;
        int secondNumber = 8;
        int sum = 0;

        sum = add(firstNumber, secondNumber);

        System.out.println(sum);
    }

    private static int add(int a, int b) {
        int returnValue = a + b;
        return returnValue;
    }
```

This is a simple function, lets break it down. We will ignore the private static for now. A function also has a return type, a name and a list of parameters. In this case, we are going to return an int, we have named the function add, and it takes 2 parameters both of type int.

TODO – Create this code and run it.

TODO – Modify the values of A and B in the add function, does that change the values of firstNumber or secondNumber?

TODO – Create a subtract function.

Some functions don’t return a value, in this case their return type is void

```
    private static void sayHello()
    {
        System.out.println("Hello");
    }
```

TODO – Create a function and call it that takes a name as a parameter, and then say Hello to that person. So if I pass in “John”, it will print “Hello, John”. Does it matter which way you call it?

```
        String myName = "John";
        sayHelloTo(myName);
        sayHelloTo("John");
```

Consider the follow program

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

    while( UserWantsToContinue() )
    {
        string calculatorString = GetCalculationFromUser();
        int result = calculateResult(calculatorString);
        printResult(result, calculatorString);
    }
}
```

Using well-named functions, it is very easy to read what this program is going to do. It also allows you as the programmer, to break up functionality into small chunks.

TODO – Create the above program and add the needed functions to make it work. Remember our work on character arrays to break up the string into two numbers and an operand. Calculator string should be something like "27-4" or "22+6". User the code below to convert a string to an int.

```
        String numberAsString = "27";
        int number = Integer.parseInt(numberAsString);
```

Use this code to build up a string from chars

```
    char digit = '5';
    char otherDigit = '1';
    numberAsString = Character.toString(digit);
    numberAsString = numberAsString + Character.toString(otherDigit);
```

TODO – Use the debugger to step through your code, how does step over (f10), step into (F11) and step out of (Shift f11) work?

TODO – Look at your calculateResult() function. Is there any code you wrote twice that you could convert to a function?

Calculator Solution

Lets walk through my solution. First your scanner class, I'll explain more later this lesson, but for now, we want a variable outside the function.

```
public class FuncCalc3 {

    static Scanner in = new Scanner(System.in);

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

You should close it with in.Close() at the end of your main though.

Next, lets look at the calculateResult function as the others should be trivial.

```
   private static int calculateResult(String calculatorString) {
        int i = 0;
        String numberAsString = "";
        char[] calculatorStringAsArray = calculatorString.toCharArray();
        int firstNumber = 0;
        char operator = ' ';
        int secondNumber = 0;

        while ( !charIsAnOperator(calculatorStringAsArray[i]) ) {
            numberAsString = numberAsString + calculatorStringAsArray[i++];
        }
        firstNumber = Integer.parseInt(numberAsString);

        operator = calculatorStringAsArray[i++];

        numberAsString = "";

        while ( i < calculatorStringAsArray.length ) {
            numberAsString = numberAsString + calculatorStringAsArray[i++];
        }
        secondNumber = Integer.parseInt(numberAsString);

        if ( operator == '+' )
            return firstNumber + secondNumber;
        if ( operator == '-' )
            return firstNumber - secondNumber;
        if ( operator == '*' )
            return firstNumber * secondNumber;
        if ( operator == '/' )
            return firstNumber / secondNumber;
        return 0;
    }

    private static boolean charIsAnOperator(char operator) {

        if ( operator == '+' )
            return true;
        if ( operator == '-' )
            return true;
        if ( operator == '*' )
            return true;
        if ( operator == '/' )
            return true;

        return false;
    }
```

A brief explanation –

1) Convert our string into a char array so we can walk the arrays
2) We have a while loop that adds each char to a string until we have an operator character.
3) We convert the string representing the first number into an int.
4) We grab the operator.
5) We build up the second number as a string and then convert to int
6) We check the operator and do the appropriate math.

TODO – Get your Calculator program working.

Switch

I don't recommend using this much, but a switch statement is some times used in place of if statements. Here is the general format. You switch on a value, and then if it matches the case, the code below it is executed. Default should always be at the bottom and is executed if nothing else matches.

```
        switch(operator ) {
            case '3':
                return false;
            case '+':
            case '-':
            case '/':
            case '*':
                return true;
            default:
                return false;
        }
```

Classes a First Look

What if we wanted a function that took the calculator string and parsed it and return the first number, the operand and the second number. Functions can only return one item, are we out of luck? This is our first look at classes, which group together information.

```
public class calculatorData {
    public int firstNumber;
    public int secondNUmber;
    public char operand;
}
```

Now we have one object that can hold 3 different pieces of information and return it all in once. Consider the following function…

```
public static calculatorData populateCalculatorData(int a, int b, char operand)
    {
        calculatorData returndata = new calculatorData();

        returndata.firstNumber = a;
        returndata.secondNUmber = b;
        returndata.operand = operand;

        return returndata;
    }
```

As always, we'll get to new, public and static. What is the return type for this function? That's right, it our class.

TODO – Add a new file called calculatorData.java and create the class above.
TODO – Modify your Calculator to have a function that parses the string and returns a calculatorData object. Then have a Calculate function that uses it.

In addition to storing like data together, classes can know how to operate and use that data. Consider the following bit of code.

```
public class CalculatorExpression3 {

    int firstNumber;
    int secondNumber;
    char operator;
    int calculate() {

        if ( operator == '+' )
            return firstNumber + secondNumber;
        if ( operator == '-' )
            return firstNumber - secondNumber;
        if ( operator == '*' )
            return firstNumber * secondNumber;
        if ( operator == '/' )
            return firstNumber / secondNumber;
        return 0;
    }
}
```

And in our main program we can call it like this.

```
CalculatorExpression3 calculatorExpression = new CalculatorExpression3();
---
int result = calculatorExpression.calculate();
```

TODO – Move the calculate and populating your class from a string into your class as functions. Your loop should now look like this.

```
        while( UserWantsToContinue() )
        {
            String calculatorString = GetCalculationFromUser();
            calculatorExpression.PopulateCalculatorExpression(calculatorString);
            int result = calculatorExpression.calculate();
            printResult(result, calculatorString);
        }
```

On Going BONUS TODO – Fizz Buzz – write a program that prints out all the numbers from 1 to 100. If a number is divisible by 3 print out Fizz instead. Divisible by 5, print out Buzz. By both, print out Fizz Buzz. (% is remainder for division, so int r = 15 % 3 gives us 0, where as 16%3 gives us 1, 17%3 gives us 2, and 18%3 gives us 0 again)