

Intro to Java

Made by your lord and savior :(

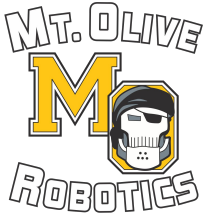
Data Types

- **int**
  - Holds integer values like 1, 2, 0, -5, -15
  - int x = 12;
  - int y = x + 16;
    - Holds 28
- **double**
  - Holds decimal values like .05, .01, -23.41
  - Can also hold integer values like 1.0, 15.0, -16.0
  - double a = 15.42;
  - double b = a * 2;
    - Holds 30.84

Data Types

- **String**
  - Holds a sequence of characters
  - “Bob”, “Hey what’s up”, “*213djaiw``dwa”
  - String str = “Cookies”;
  - String str2 = “Snickerdoodle” + str;
    - Holds “Snickerdoodle Cookies”
- **boolean**
  - Holds true or false
  - boolean bool = true
  - boolean bool2 = myAge > 5 && myAge < 10;
  - boolean bool3 = myAge < 18 || myAge > 60;

If Statements

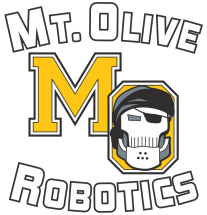



```
int myAge = 17;
int brothersAge = 81;
String myName = "Bob";

if(myAge < brothersAge){
      System.out.println("You are younger than your brother");
}else if (myAge > brothersAge){
      System.out.println("You are older than your brother!");
}else if (myAge == brothersAge){
      System.out.println("You are the same age as your brother");
}

if (myName.equals("Joe")){
      System.out.println("Your name is Joe!");
}else{
      System.out.println("Your name is not Joe!!!");
}
```

Java Operators



- >, >=
  - Greater than, greater than or equal
- <, <=
  - Less than, less than or equal
- ==
  - Equal to
- !=
  - Is not equal to
- &&
  - And (followed by another condition)
- ||
  - Or (followed by another condition)

More Operators

- +
  - Adds two values
- -
  - Subtracts two values
- *
  - Multiplies two values
- /
  - Divide two values
- %
  - Gives you the remainder of the first value divided by the second

Loops

While loop

```
int i = 0;
while (i < 100){
      System.out.println(i);
      i++;
}
//Prints out numbers 0-99
```

While loop outline:

```
while(condition == true)
```

Loops

For-Loop

```
for(int i = 0; i < 100; i++){
      System.out.println(i);
}

//Prints out numbers 0-99
```

For loop outline:

for (initialization; conditions to keep loop running; increment/decrement)

The Main Method

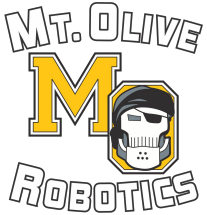
Only code in the main method is going to run!!!

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

Fizz*Buzz*

- Use a for-loop to loop through numbers 0-100
- For every multiple of 3, print out “Fizz”
- For every multiple of 5, print out “Buzz”
- For every multiple of 3 and 5, print out “FizzBuzz”
- If not a multiple of 3 or 5, just print out the number

```java
public static void main(String[] args){
     for(int i = 0; i <= 100; i++){
          if(i % 3 == 0 && i % 5 == 0){
               System.out.println("FizzBuzz");
          }else if (i % 3 == 0)
               System.out.println("Fizz");
          }else if(i % 5 == 0){
               System.out.println("Buzz");
          }else{
               System.out.println(i);
          }
     }
}
```



Taking User Inputs

- A scanner allows us to get a user input

```
public static void main(String[] args){
     Scanner input = new Scanner(System.in);
     System.out.println("Enter an integer");
     int userNum = input.nextInt();
     System.out.println("Enter a decimal number");
     double userDecimal = input.nextDouble();
     System.out.println("Enter a string");
     String userString;
     userString = input.nextLine();
}
```

Calculator??

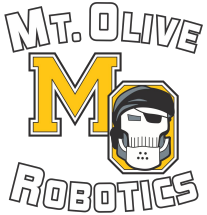
What do you need?

- double num1, num2;
  - Program should ask the user what numbers they want to use
    - Remember the Scanner class!!
- String operation
  - Program should ask the user what operation they want to do
  - The user should enter either “Addition”, “Subtraction”, “Multiplication”, or “Division”
- If statements to compare the operation
  - Did the user input Addition? Subtraction? Multiplication? Division
  - Do the selected operation with the num1 and num2
  - Print out the result

Methods

- Breaks up our code into smaller parts!
  - All our code doesn't need to go into the main method!
    - Makes our code more easy to read
  - Each method should perform a small, concise, and often repeated action
  - If you need to use the same piece of code more than once, make a method!
- Void methods
  - ```
    public void sayHello(){
        System.out.println("Hello World!");
    }
    ```
- Return methods
  - ```
    public int getAge(){
        return 16;
    }
    ```
  - ```
    public String getName(){
        return "Vedh";
    }
    ```
    ```

More Complex Examples



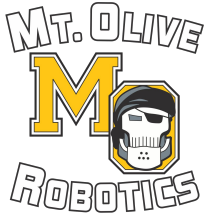

```
public double getCircleArea(int radius){
    return Math.PI * Math.pow(radius, 2)
}

public String getFirstFiveLetters(String str){
    return str.getSubstring(0, 5)
}
```

`int radius` and `String str` are called ***arguments***

- Extra information that a method needs
- ***Arguments*** are specified in the methods parentheses

Calling Methods

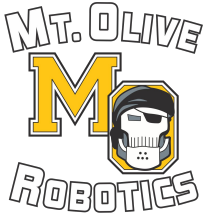



```
public class MyClass{
     public static void main(String[] args){
          sayHello();
          double area = getCircleArea(10.51);
          System.out.println(area);
     }

     public void sayHello(){
          System.out.println("Hello!");
     }

     public double getCircleArea(double radius){
          return Math.Pi * Math.pow(radius, 2);
     }
}
```

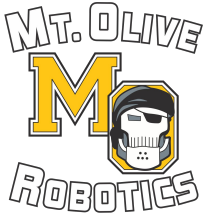
Classes and Objects!!!!!!




```
public class Dog{
    String name = "Aruuuuuuuu";
    String type = "Husky";
}

public class Cat{
    String name = "Joe";
    String type = "Lion";
}
```

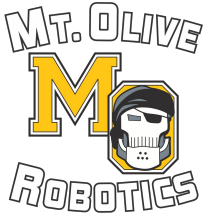
Classes



Both “Dog” and “Cat” are called classes

- Classes can have their own unique data and variables
- Classes can have their own unique methods
- What if we want to reference the variables and methods in one class from a completely separate class?
    - AWESOME OBJECTS!!!!

Objects




```java
public class MyClass{
	public static void main(String[] args){
		Person p1 = new Person();
		p1.setName("Bob");
		System.out.println(p1.name);
		p1.increaseAge();
		System.out.println(p1.age);

		Person p2 = new Person();
		System.out.println(p2.name);
	}
}
```

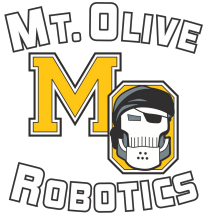
- Objects are created by writing:
  - ClassName anyVariableName = new ClassName();

```java
public class Person {
	String name = "NO NAME SET";
	int age = 0;

	public void setName(String newName){
		name = newName;
	}

	public void increaseAge(){
		age++;
	}
}
```

CONSTRUCTORSSSSS!!!




```
public class MyClass {
      public static void main(String[] args){
            Person p1 = new Person("Joe", 42)
            System.out.println(p1.name);
            System.out.println(p1.age);
      }
}
```

- Constructors are useful for providing initial data that objects often need (i.e. name and age in the Person class)
- Constructor is called as soon as your object is created
    - IS NOT A METHOD!! ONLY RUNS ONCE WHEN YOUR OBJECT IS CREATED

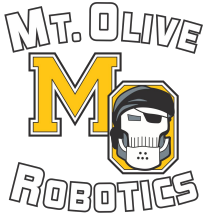
```
public class Person{
      String name;
      int age;

      public Person(String personName, int personAge){
            name = personName;
            age = personAge;
      }

      public void setName(String newName){
            name = newName;
      }

      public void increaseAge(){
            age++;
      }
}
```

Public vs. Private

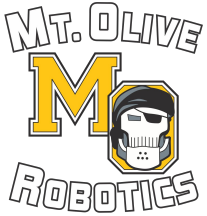



```
public class Person {
	public String name;
	public int age;
	private String socialSecurityNumber;
}
```

- **Person myPerson = new Person();**
  - I can always write ***System.out.println(myPerson.name)*** and ***System.out.println(myPerson.age)*** without getting an error
  - I can **not** write ***System.out.println(myPerson.socialSecurityNumber)*** without getting an error
    - ***socialSecurityNumber*** is labeled as “private”, which means objects like ***myPerson*** can not access them
      - ➢ You can still edit private variables like “socialSecurityNumber” inside the Person class, just not in any other class

Accessing/Modifying Private Variables

Getters and Setters!!!!




```
public class MyClass{
	public static void main(String[] args){
		Person p1 = new Person("Bob", "2321-333-2134", 55);
		System.out.println(p1.getSocialSecurityNumber());
		//Outputs 2321-333-2134
		p1.setSocialSecurityNumber("4213-123-4551”);
		System.out.println(p1.getSocialSecurityNumber());
		//Outputs 4213-123-4551
	}
}
```

```
public class Person {
	public String name;
	private String socialSecurityNumber;
	public int age;
	public Person(String personName, String SSN, int
		personAge){
		name = personName;
		socialSecurityNumber = SSN;
		age = personAge;
	}

	public void setSocialSecurityNumber(String newSSN){
		socialSecurityNumber = newSSN;
	}

	public String getSocialSecurityNumber(){
		return socialSecurityNumber;
	}
}
```

Make me a Bank

Bank Account Class

- Needs three private variables
  - Name of Account Holder (String)
  - Account PIN (String)
    - 4 Digit PIN
  - Balance (double)
- Constructor
  - Takes in two strings (Name and PIN) and a double (initial deposit for balance variable)
    - Set instance variables equal to these arguments
- Methods
  - Getters and setters for each instance variable
  - Deposit and Withdraw method
    - Takes in a double as an argument and deposits/withdraws accordingly

Make me a Bank

```
Successfully logged in as: Bob
Welcome Bob! Your current balance is 500.0! Select an option
1. Withdraw
2. Deposit
3. Close account
4. Log out
```

Tester Class

- Create one bank account object
- Have the user input a pin (Scanner class!!)
- Check if the pin matches the pin on the bank account object
  - If yes, print out a nice looking menu that gives the user an option to Withdraw, Deposit, or log out
  - If the user asks to withdraw or deposit money, ask them how much they would like to withdraw/deposit
    - Subtract or add from/to balance variable accordingly
  - If the user asks to do nothing, then just print out “Logging Out”