Brief Overview

How much do we remember?

Review: Variables

- Variables are created to hold data values to be manipulated throughout the code with conditions or loops.
- The commonly used primitive data types, are:

● Integer (int) ● Double (double) ● String (String)	● Float (float) ● Character (char) ● Boolean (boolean)

Syntax: Variables

- The general syntax for variables are:
  public/private data type “variable name” = specified data type value;

- Ex: public int numKittens = 5;
  private double area = 24.8;
  public String greeting= “Hello projammers”;

Review: Conditionals

- Conditionals are statements that check if a given condition is true. If true, compiler executes code written inside or the code is skipped if evaluated to false.

- Conditionals are identified with the keywords: “if”, “else if”, and “else”

Syntax: Conditionals

- The syntax of Conditionals are:

```
if/else/else if (condition){
    [code to be
    executed];
}
```

- Ex: if (age > 18 && license = true){

```
        System.out.println("I can drive!");
    }
```

Exercise:

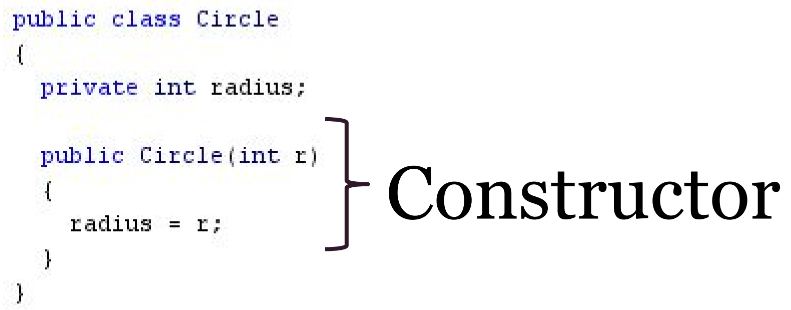
- Create two integer variables: one to denote the total amount of cookies and another to show how many cookies I ate. For every cookie I eat, the total amount of cookies decrease by one and the cookies I eat increases by 1. Once I eat 10 cookies, have the compiler print out: “I am too full!” and the total number of cookies left.

Review: Classes

- Classes list the attributes of certain objects and actions that objects under this class can perform.
- Classes increase encapsulation, allowing us to hide certain properties in code, which ensures variables will not be easily changed by another programmer.

Syntax: Classes

- When declaring classes, each class comes with a default constructor. Constructors ALWAYS have the same name as the class.
- The constructor specifies the attributes of the object created under the class.
- Ex:



```
public class Circle
{
  private int radius;

  public Circle(int r)
  {
    radius = r;
  }
}
```

Syntax: Classes

- Declaring a class requires you to name your class (capitalized by convention):

  public/private class “CapitalizedName”

- Ex: public class CountTheCats

  private class Jarvis

Review: Methods

- Methods allow objects to perform an action. Methods can either return a value of a specified data type or simply calculate/change a given variable.

- Methods can also take parameters that are manipulated in the method.

Syntax: Methods

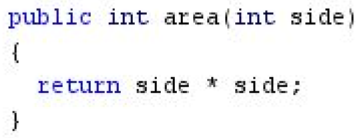
- Methods are usually named using verbs to describe the action they complete:

  $\boxed{\text{public/private}}$ <u>data type</u> “Method name” (parameters) {

  [executable code]

  }

- Ex:



```
public int area(int side)
{
  return side * side;
}
```

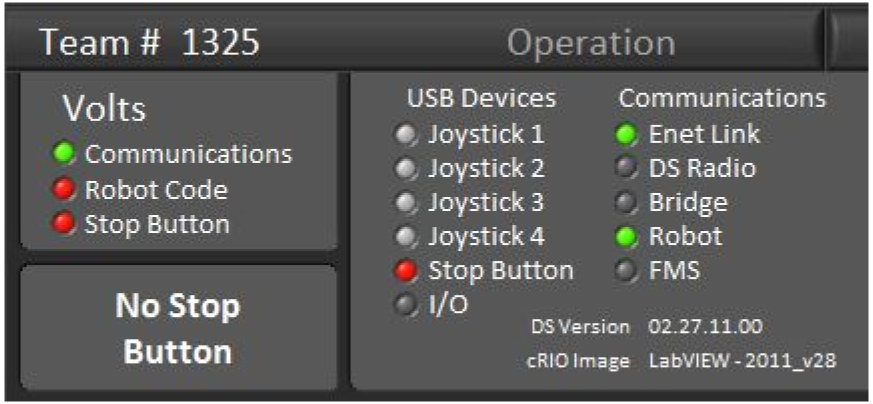
Practice with Methods!

The link below contains exercises to practice and get comfortable with methods:

Methods Practice

From code to robot

- Once our robot code is completed, we deploy and load our code onto a program called the **Driver Station**.



Team # 1325 Operation

Volts
- Communications
- Robot Code
- Stop Button

No Stop Button

USB Devices
- Joystick 1
- Joystick 2
- Joystick 3
- Joystick 4
- Stop Button
- I/O

Communications
- Enet Link
- DS Radio
- Bridge
- Robot
- FMS

DS Version 02.27.11.00
cRIO Image LabVIEW - 2011_v28

From code to robot

- Robot code references specific ports on the actual robot where motors and sensors are connected to.
- These ports are denoted in our code with constant variables under a class called the RobotMap.

Common Robot Lingo

- RoboRIO: The “brain” of the robot. This part receives all of our code and is where all electronics parts are connected into numbered ports.
- Motors: We use motors called Talons which are wired to any motorized part on the robot and connected to the “brain” of the robot.
- Solenoid: Parts attached to pistons which control the piston’s ability to extend and retract.
- Pistons: Empty canisters that fill up with air from the air compressor.

Robot Sensors

- Gyro: A small chip attached on the robot to sense the angle at which the robot has rotated relative to its starting position as North (0°)
- Camera: Usually a webcamera, this part is mounted on the robot to receive vision from the robot's POV or to calculate the amount of pixels from reflective tape on field objects.
- Encoders: Sensors that measure the rotation of a spinning wheel which is used to measure the distance the robot has traveled. Device measured in ticks.

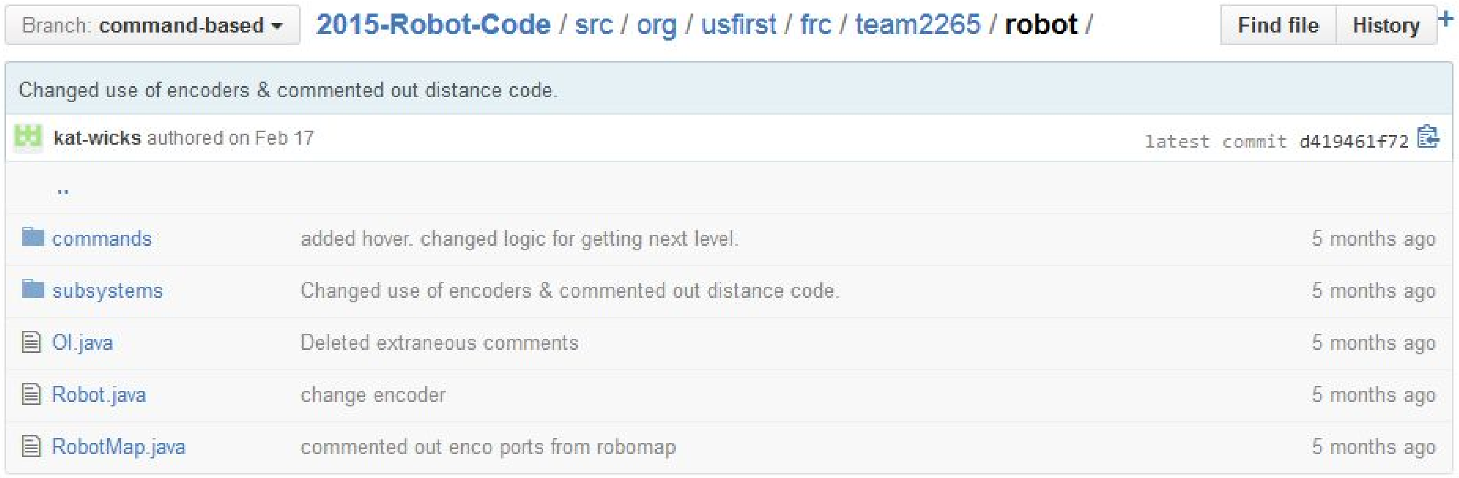
Types of Coding Methods

There are multiple ways to write robot code. We have switched between two types:

- IterativeRobot: Code in this format is written into one class with all the methods and variables in one place.
- Command Based: Code in this format is broken into 3 categories of classes: Subsystems, Commands, and Command Groups.

Command Based

Command Based Programming is often used because it promotes an organized and efficient method of coding. Examples from our command based code look like this:



2015-Robot-Code / src / org / usfirst / frc / team2265 / robot /

Changed use of encoders & commented out distance code.

kat-wicks authored on Feb 17 latest commit d419461f72

..

commands	added hover. changed logic for getting next level.	5 months ago
subsystems	Changed use of encoders & commented out distance code.	5 months ago
OI.java	Deleted extraneous comments	5 months ago
Robot.java	change encoder	5 months ago
RobotMap.java	commented out enco ports from robomap	5 months ago

Categories of Command Based

- Subsystems: Refers to parts of the robot that are controlled independently and outlines a set of methods that the part can complete.
- Commands: Calls methods from existing subsystems and executes them in a specified order to complete a task.
- Command Groups: Allows for the robot to execute to actions at the same time. (ex: shoot an object while the robot picks up the next game piece).

Homework

When writing our robot code, the API is our greatest friend in searching up methods and what they do. Here is the link to the API:

API

- With this API, write the method we would use to create a TankDrive(). (this allows our robot to drive!).