



# More Than Working Code

MRA Education Day 2018

## Did you have breakfast this morning?

Yes, of course. Breakfast is the most important meal of day.

No, didn't have time.

Nope, don't usually eat breakfast.

Breakfast is for the weak.

# Listener Design Pattern

## Updating Sensor

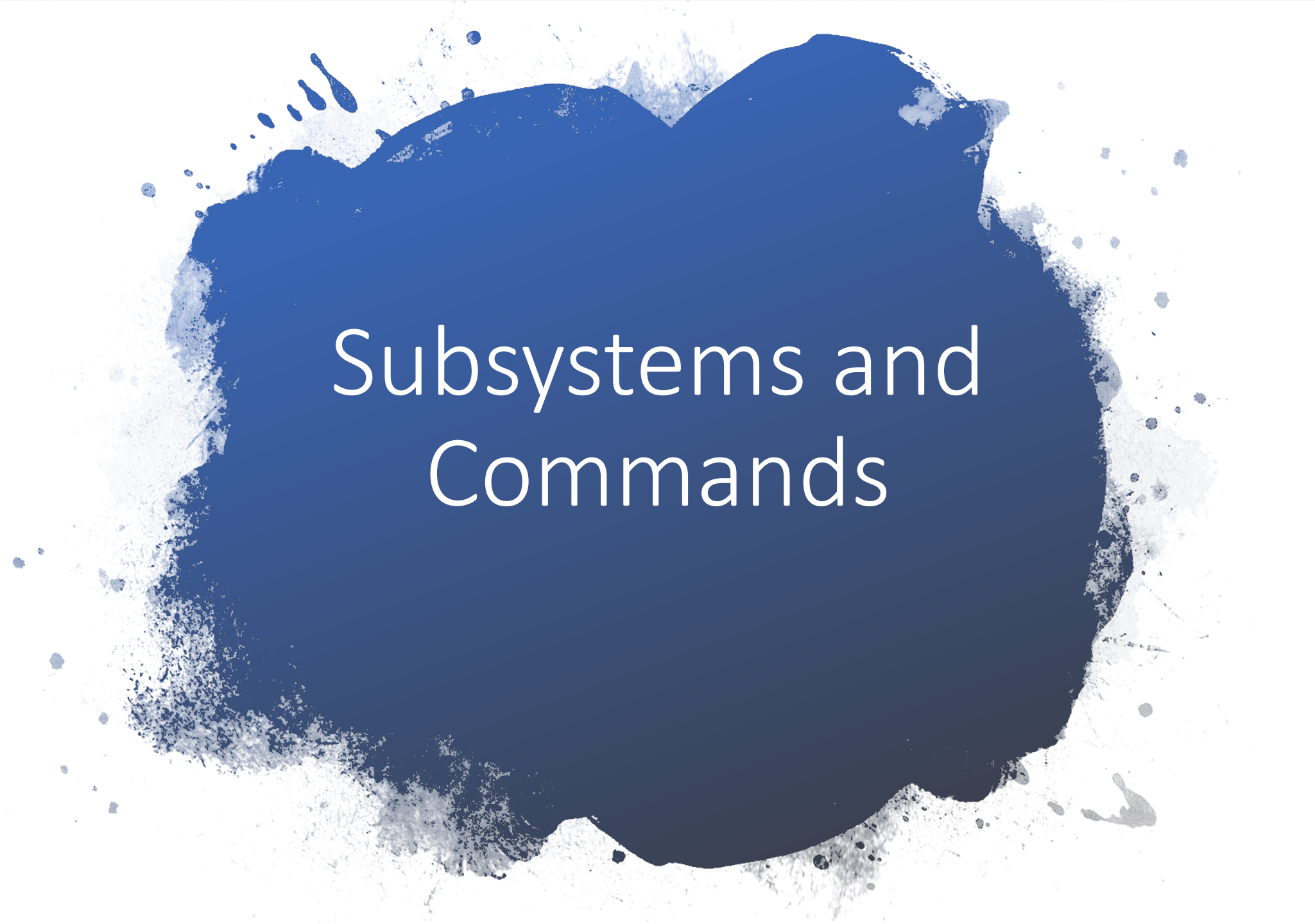
- Sends out Event objects to anyone who asks

## Listener

- Receives new Event objects from the source

## Events

- Contains the relevant information about the change in some sensor



# Subsystems and Commands

# What is a Subsystem

- A subsystem is a part of the robot that works independently of the rest
- Typically used to model a mechanical system on the robot
- Contains methods and variables used to interface with the mechanical system
- Associated with one or more commands

# Examples of Common Subsystems

- Drive Subsystem
- Climber Subsystem
- Arm/Manipulator Subsystem

# Commands

- Associated with a subsystem
- Uses that subsystem to perform an action
- Commands may be registered to run when a button is pressed, released, or held

# The Default Command

- Every Subsystem can specify 1 default command
- This command runs when no other command associated with the subsystem is running
- If any other command associated with the subsystem starts the default command is interrupted



# Structure of a Command



initialize - runs once when command starts



execute - runs every 20 milliseconds while the command is running



isFinished - runs every 20 milliseconds after execute

Returns true if the command should exit and return to the default command

Returns false otherwise



end - runs once if the command ends peacefully (isFinished returns true)



interrupted - runs once if the command is interrupted

For example, if the button of a whileHeld command is released

## Example Subsystem (DriveSubsystem.java)

```
public class DriveSubsystem extends Subsystem {
    private TalonSRX leftWheel;
    private TalonSRX rightWheel;

    public DriveSubsystem() {
        leftWheel = new TalonSRX(Ports.FRONT_LEFT_DRIVE_MOTOR);
        rightWheel = new TalonSRX(Ports.FRONT_RIGHT_DRIVE_MOTOR);
    }

    @Override
    public void initDefaultCommand() {
        this.setDefaultCommand(new DriveCommand());
    }

    public void setMotors(double speed) {
        leftWheel.set(speed);
        rightWheel.set(speed);
    }
}
```

## Example Command (DriveCommand.java)

```
public class DriveCommand extends Command {
    public DriveCommand() {
        requires(Robot.driveSys);
    }
    // We don't need to initialize anything for this command
    protected void initialize() {}

    protected void execute() {
        Robot.driveSys.setMotors(.5); // sets both motors to 50% speed
    }

    protected boolean isFinished() { return false; } // always runs

    protected void end() {
        Robot.driveSys.setMotors(0);
    }

    protected void interrupted() {
        end(); // stop motors when finished
    }
}
```

## Initializing a Subsystem (Robot.java)

```
public static DriveSubsystem driveSys;
```

```
@Override
```

```
public void robotInit() {
```

```
    driveSys = new DriveSubsystem();
```

```
    driveSys.initDefaultCommand();
```

```
}
```

```
@Override
```

```
public void teleopPeriodic() {
```

```
    // All subsystems automatically register themselves with the scheduler
```

```
    // So all you have to do is call Scheduler.getInstance().run()
```

```
    // Every loop in teleopPeriodic
```

```
    Scheduler.getInstance().run();
```

```
}
```

## Assigning a Command to a Button

```
public Joystick rightJoystick = new Joystick(Ports.RIGHT_JOYSTICK);  
public Button driveStartButton =  
    new JoystickButton(rightJoystick, Ports.DRIVE_START_BUTTON);  
driveStartButton.whenPressed(new DriveManualCommand());
```

# Command Groups

- Run multiple commands at a time or in sequence
  - Add commands in the constructor
- Called exactly like regular commands are
- Useful for autonomous mode or automation in general

# Command Group Example

```
public class SameSideScaleRoute extends CommandGroup {  
    public SameSideScaleRoute(boolean left){  
        addSequential(new DriveStraightCommand(345));  
        if(left) addSequential(new RotateCommand(90));  
        else addSequential(new RotateCommand(-90));  
  
        addParallel(new LowerFlipperCommand(), Specs.FLIPPER_SCALE_LOWER_TIME);  
        addParallel(new DriveStraightCommand(-40), 0.69);  
        addSequential(new VertUpCommand(Specs.SCALE_HEIGHT));  
        addSequential(new ExpelCommand(1), Specs.EXPEL_TIME);  
    }  
}
```

# Further Reading

<http://wiki.team2537.com/wiki/SoftwareTrainingIntermediate>



The image features a dark gray background with a central white horizontal band. Behind the band, three overlapping blue circles are visible, creating a pattern of lighter blue and white intersections. The text "Break for Some Comics" is centered within the white band.

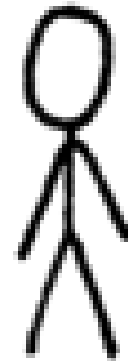
Break for Some Comics

MAKE ME A SANDWICH.

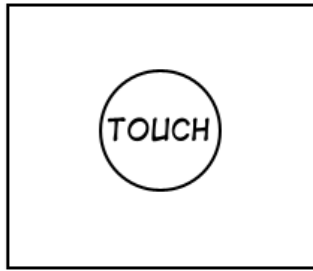
WHAT? MAKE  
IT YOURSELF.

SUDO MAKE ME  
A SANDWICH.

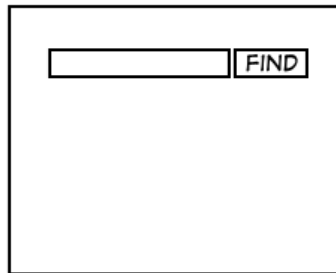
OKAY.



TYPICAL APPLE PRODUCT...



A GOOGLE PRODUCT...



YOUR COMPANY'S APP...

FIRST NAME: <input type="text"/>	TYPE CD: <input type="text"/>	4 - K
LAST NAME: <input type="text"/>	TQP STAT: <input type="checkbox"/>	AA2-
SSN: <input type="text"/>	VER: <input type="text"/>	DK9B
ID: <input type="text"/>	FT/PT: <input checked="" type="checkbox"/>	KKA?
PHONE 1: <input type="text"/>	CAT CD: <input type="text"/>	CN3
PHONE 2: <input type="text"/>	CITY: <input type="text"/>	AA-9
ADDR 1: <input type="text"/>	STATE: <input type="text"/>	NEW
ACCT #: <input type="text"/>	ZIP: <input type="text"/>	DEL
	ORD #: <input type="radio"/> <input type="radio"/> <input type="radio"/> ? <input type="radio"/>	
OKAY APPLY SAVE UNDO HELP DELETE EDIT		
SELECT BROWSE ERRORS		

THE #1 PROGRAMMER EXCUSE  
FOR LEGITIMATELY SLACKING OFF:

"MY CODE'S COMPILING."

HEY! GET BACK  
TO WORK!

COMPILING!

OH. CARRY ON.

HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.



OH, DEAR - DID HE BREAK SOMETHING?

IN A WAY-



DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?



OH, YES. LITTLE BOBBY TABLES, WE CALL HIM.

WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.



AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

# SQL Injection

---

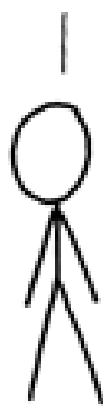
nano? REAL PROGRAMMERS USE emacs



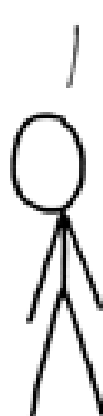
HEY. REAL PROGRAMMERS USE vim.



WELL, REAL PROGRAMMERS USE ed.



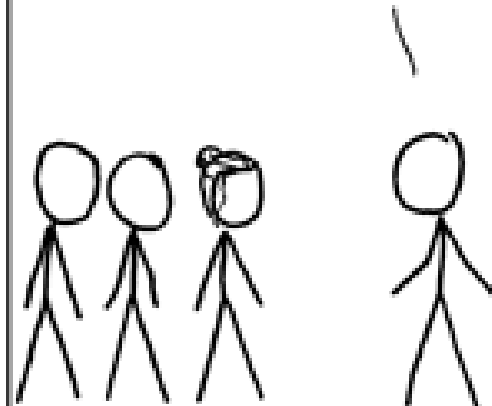
NO, REAL PROGRAMMERS USE cat.



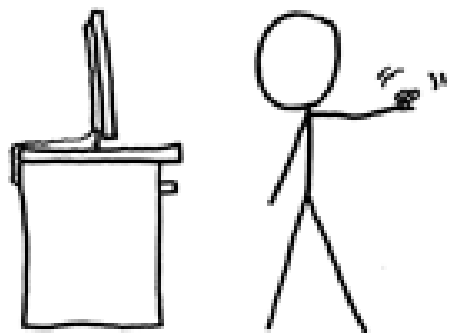
REAL PROGRAMMERS USE A MAGNETIZED NEEDLE AND A STEADY HAND.



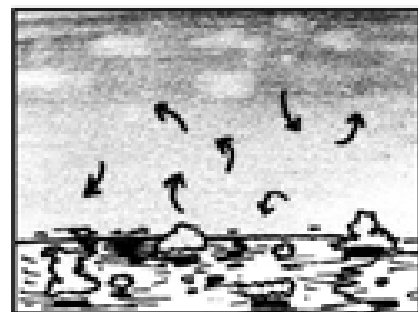
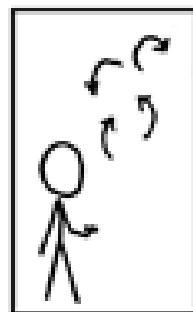
EXCUSE ME, BUT REAL PROGRAMMERS USE BUTTERFLIES.



THEY OPEN THEIR HANDS AND LET THE DELICATE WINGS FLAP ONCE.

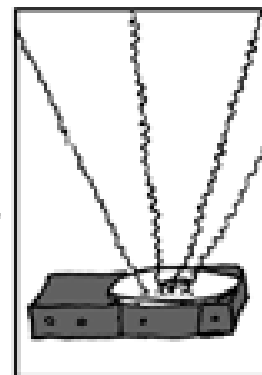
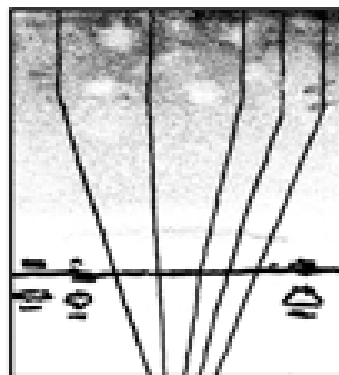


THE DISTURBANCE RIPPLES OUTWARD, CHANGING THE FLOW OF THE EDDY CURRENTS IN THE UPPER ATMOSPHERE.



THESE CAUSE MOMENTARY POCKETS OF HIGHER-PRESSURE AIR TO FORM,

WHICH ACT AS LENSES THAT DEFLECT INCOMING COSMIC RAYS, FOCUSING THEM TO STRIKE THE DRIVE PLATTER AND FLIP THE DESIRED BIT.



NICE. 'COURSE, THERE'S AN EMACS COMMAND TO DO THAT.

OH YEAH! GOOD OL' C-x M-c M-butterfly...



DAMMIT, EMACS.



OO Design

Principle of  
Least  
Privilege

*They don't have  
to know...*

“TIM – don't look  
below this line”

# Cohesion vs. Coupling

## Coupling

- How well each part stands on its own
- Increase

## Coupling

- How classes are related to each other
- Generally decrease



Open-  
Closed  
Principle

Open for  
extension

Closed for  
modification



# Control Theory

# Control Structures

## PID Loops

- Proportional – based on current distance away from target
- Integral – based on how long (and how far) it's taken to get to target so far
- Derivative – based on how fast it's currently approaching

## More advanced structures (aimed for high schoolers):

- Source: <https://github.com/calcmogul/state-space-guide>
- PDF: <https://file.tavsys.net/control/state-space-guide.pdf>



# How to Get Better Documentation?

Let's Discuss