

# FRC Java 2010

- Java basics
- Development environment
- Features/comparison with C++
- Our code
- Virtual Machine information
- Questions

# Java Basics

- High-level, object-oriented, portable, and fast.
- Uses both compilation and a virtual machine to run code.
- Designed for embedded systems, but is common with personal computers and online software.

# NetBeans IDE

- This is our development environment for Java.
- Programmed in Java itself, it is designed for Java programming but can do other languages (such as C/C++)
- Uses plugins to connect to the cRio
- Has great debugging capabilities, as well as good code-completion tools.

RobotTemplate.java \* x

```

double[] right= { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0 };
for (int i=0; i<15; i++)
{
    drivetrain.tankDrive(left[i], right[i]);
    Timer.delay(1.0);
}
drivetrain.tankDrive(0.0, 0.0);
}

/**
 * This function is called once each time the robot enters operator control.
 */
public void operatorControl()
{
    while(true)
    {
        drivetrain.arcadeDrive(leftStick, false);
        if (rightStick.getTrigger())
            launcher.set(Relay.Value.kReverse);
        else
            launcher.set(Relay.Value.kOff);
        if (rightStick.getRawButton(3))
            conveyor.set(1.0);
        else
            conveyor.set(rightStick.getY());

        // Begin new crab drive code

        lspwr = 0.0;
        rspwr = 0.0;
        if (leftStick.getRawButton(6))
            if (!leftStick.getRawButton(7))
                // Assuming:
                // 6 11
                // 7 10
                lspwr = 1.0;
        if (leftStick.getRawButton(7))
            if (!leftStick.getRawButton(6))
                lspwr = -1.0;
        if (leftStick.getRawButton(11))
            if (!leftStick.getRawButton(11))
                rspwr = 1.0;
        if (leftStick.getRawButton(10))
            if (!leftStick.getRawButton(11))
                rspwr = -1.0;
        leftsteer.set(lspwr);
        rightsteer.set(rspwr);

        Timer.delay(0.005);
    }
}
}

```

Calculator  
CelsiusConverterProject  
language  
Robot  
src  
edu.wpi.first.wpilibj.ten  
RobotTemplate.java  
build.xml

operatorControl -... x

Members View

RobotTemplate :: Simple  
 RobotTemplate()  
 autonomous()  
 operatorControl()  
 conveyor : Victor  
 drivetrain : RobotDrive  
 launcher : Relay  
 leftStick : Joystick  
 leftdrive : Victor  
 leftsteer : Victor  
 lspwr : double  
 rightStick : Joystick  
 rightdrive : Victor  
 rightsteer : Victor  
 rspwr : double

87:40 INS

Tasks Output

# Comparison with C++

- Java
- NetBeans
- Virtual machine
- Garbage collector
- References
- Packages
- Single inheritance with interfaces
- Runtime error checking
- Lower performance (at least on the cRIO.)
- Names like `getValue()`
- Utility functions, like a wait, are methods, like `Timer.delay(1.0);`
- C++
- Windriver
- Compiled
- Manual memory freeing
- Pointers
- Includes/preprocessor
- Multiple inheritance
- No runtime error checks
- Highest performance (machine code)
- Names like `GetValue()`
- Utility functions, like just saying `Wait()`

# Our Code

- Uses basic template
- Runs a simple autonomous, launcher/conveyor, arcade drive, and steering motor power
- Much more compact than our Labview-based code from last year.
- Has initialization, teleop mode, and autonomous functionality.

```
package edu.wpi.first.wpilibj.templates;  
import edu.wpi.first.wpilibj.RobotDrive;  
import edu.wpi.first.wpilibj.SimpleRobot;  
import edu.wpi.first.wpilibj.Timer;  
import edu.wpi.first.wpilibj.Joystick;  
import edu.wpi.first.wpilibj.Relay;  
import edu.wpi.first.wpilibj.Victor;
```

```
public class RobotTemplate extends SimpleRobot
{
    private Victor leftdrive;
    private Victor rightdrive;
    private Victor leftsteer;
    private Victor rightsteer;
    private RobotDrive drivetrain;
    private Joystick leftStick;
    private Joystick rightStick;
    private Relay launcher;
    private Victor conveyor;
    private double lspwr;
    private double rspwr;
```

```
public RobotTemplate()
{
    leftdrive = new Victor(1);
    rightdrive = new Victor(2);
    drivetrain = new RobotDrive(leftdrive, rightdrive);
    leftStick = new Joystick(1);
    rightStick = new Joystick(2);
    launcher = new Relay(1);
    conveyor = new Victor(6);
    leftsteer = new Victor(3);
    rightsteer = new Victor(4);

    drivetrain.setInvertedMotor(RobotDrive.MotorType.kRearLeft, false);
    drivetrain.setInvertedMotor(RobotDrive.MotorType.kRearRight, false);
}
```

```
public void autonomous()
{
    double[] left = { 1, 1, 1, 0, 0, 0, -1, -1, -1, 1, 1, 1, 1, 1, 1 };
    double[] right= { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0 };
    for (int i=0; i<15; i++)
    {
        drivetrain.tankDrive(left[i], right[i]);
        Timer.delay(1.0);
    }
    drivetrain.tankDrive(0.0, 0.0);
}
```

```
public void operatorControl()
{
    while(true)
    {
        drivetrain.arcadeDrive(leftStick, false);
        if (rightStick.getTrigger())
            launcher.set(Relay.Value.kReverse);
        else
            launcher.set(Relay.Value.kOff);
        if (rightStick.getRawButton(3))
            conveyor.set(1.0);
        else
            conveyor.set(rightStick.getY());

        // Begin new crab drive code.
```

```
lspwr = 0.0;
rspwr = 0.0;
if (leftStick.getRawButton(6))
    if (!leftStick.getRawButton(7))
        lspwr = 1.0;
if (leftStick.getRawButton(7))
    if (!leftStick.getRawButton(6))
        lspwr = -1.0;
if (leftStick.getRawButton(11))
    if (!leftStick.getRawButton(10))
        rspwr = 1.0;
if (leftStick.getRawButton(10))
    if (!leftStick.getRawButton(11))
        rspwr = -1.0;
leftsteer.set(lspwr);
rightsteer.set(rspwr);

Timer.delay(0.005);
}
}
```

# Virtual Machine

- Squawk VM, based on Java ME – a simplified version of Java for embedded devices
- Dynamic class loading or unloading and reflection are not supported
- Pre-verification is done.
- No finalization
- No Java Native Interface, but JNA is similar
- No Serialization and Remote Method Invocations
- No UI APIs
- Different thread classes
- Based on Java SE 1.3, so no Generics, Annotations, or Autoboxing