# INTRODUCTION TO ROBOTICS PROGRAMMING

Team 294

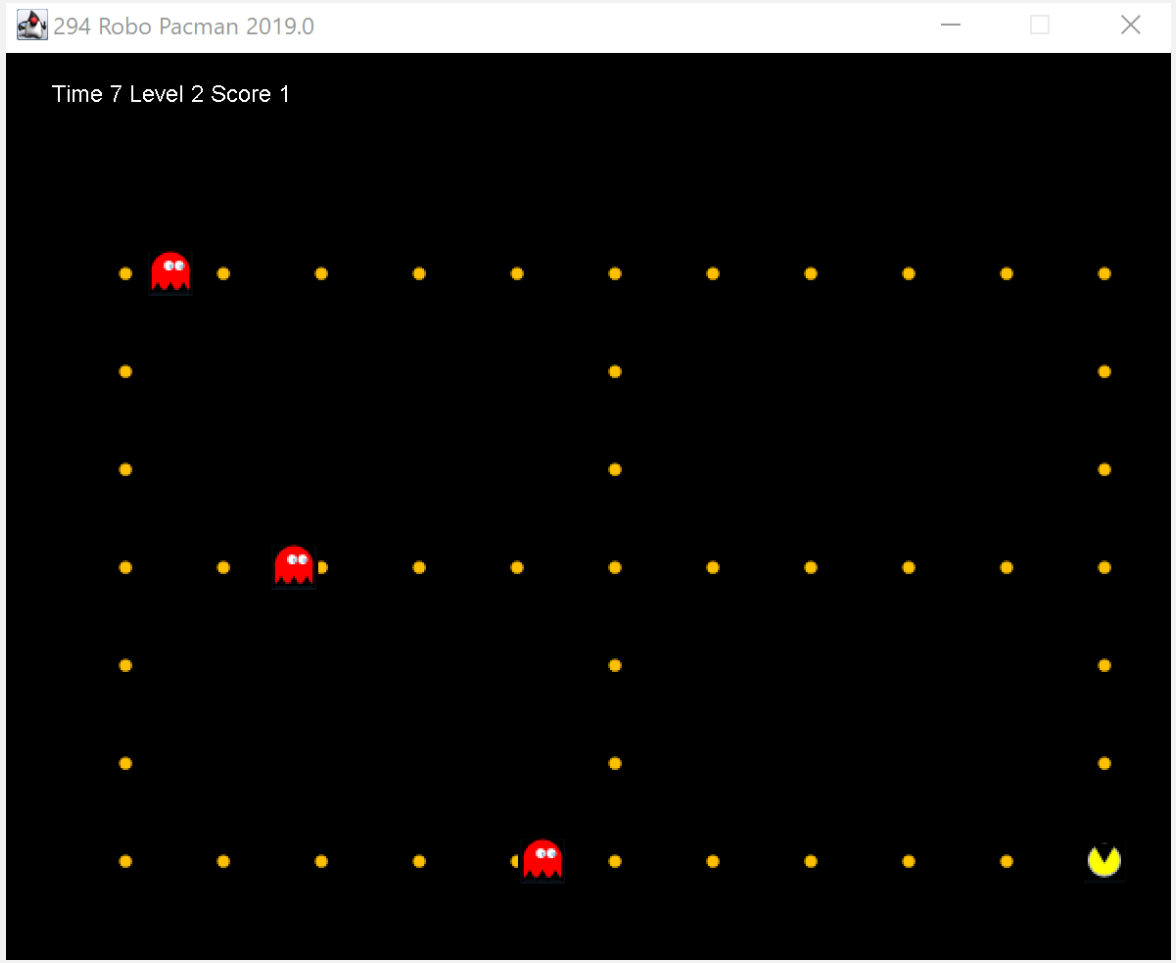
# GOALS

- Introduce FRC style programming with Java
- Learn how to use the development tools (Visual Studio Code and Github)
- Practice programming on a virtual robot
- Prepare for the tryout

# ROBO PACMAN

Robot simulator

- Command based programming
- Autonomous
- Tank drive
- Dot sensor
- Ghost sensor



# DRIVETRAIN

Tank drive

- tankDrive(double left, double right)
- int getDistance()
- int getAngle()

0 is north

270 is west 90 is east

180 south

Example: Robot.driveTrain.tankDrive(1, 1)

# COMMAND BASED PROGRAMMING

When the Robot runs it executes a series of commands

Command groups control the order

The scheduler repeatedly calls each command until the command is finished and then moves on to the next

Can be executed sequentially or in parallel in the real robot but Pacman only supports sequential

```
1   package pacman.commands;
2
3   import pacman.base.CommandGroupBase;
4
5   public class AutoGroup extends CommandGroupBase {
6
7       public AutoGroup() {
8           addSequential(new DriveStraight(200));
9           addSequential(new Turn(90));
10          addSequential(new DriveStraight(200));
11          addSequential(new Turn(0));
12          addSequential(new DriveStraight(200));
13      }
14
15  }
16
```

# COMMANDS

Commands should be reusable and designed to be grouped together to achieve a goal

- Turn
- DriveStraight
- PickupBall

Extends CommandBase

- void initialize()
- void execute()
- boolean isFinished()

```java
package pacman.commands;

import pacman.base.CommandBase;
import pacman.robot.Robot;

public class SpinForever extends CommandBase {

    protected void execute() {
        super.execute();

        System.out.println("Hello world, watch me spin to the right");
        Robot.driveTrain.tankDrive(1, 0);

    }


}

```

# TYPICAL SEQUENCE

1. init()
2. execute()
3. isFinished() returns false
4. execute()
5. isFinished() returns false
6. execute()
7. isFinished() returns true

# SETUP

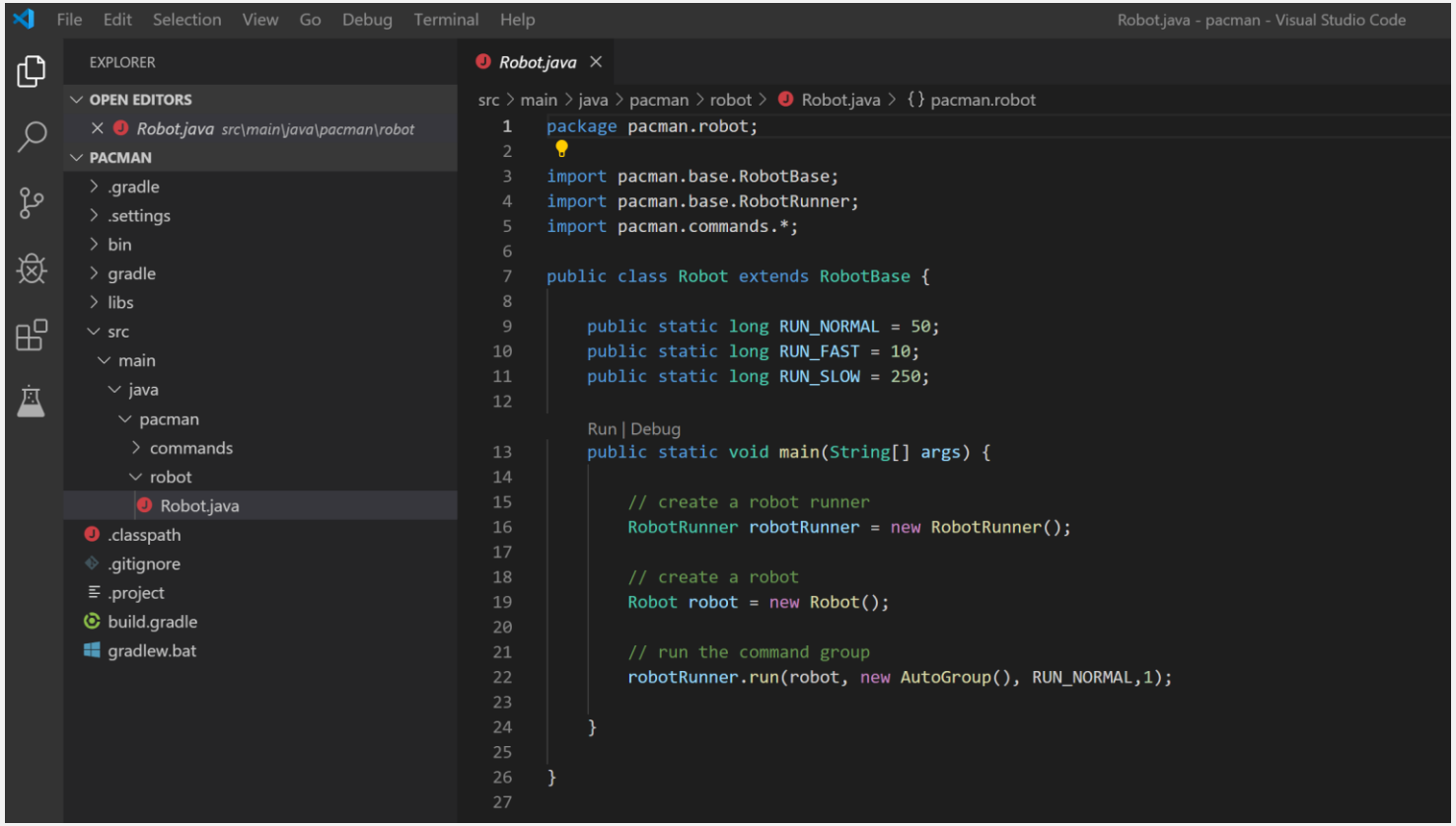
- Windows 10 is used in the lab and during competition
  - MacOS or Linux is also fine for Robo Pacman
- Java Development Kit (JDK SE 11 or higher)
  - Best to use the same one that came with WPILIB (C:\Users\Public\wpilib\2020\jdk)
- Visual Studio Code will be used in lab but any IDE will work
  - https://code.visualstudio.com/download
  - https://code.visualstudio.com/docs/java/java-tutorial
  - Set JAVA_HOME system environment variable or configure java.home in VS Code to point to JDK directory from above
  - In VS Code, use Control Palette Java: Configure Java Runtime

# INSTALL

- Get the code from Github
  - Clone https://github.com/team294/RoboPacman
- Open folder in Visual Studio Code
  - C:\Users\Paul\Documents\GitHub\RoboPacman\pacman

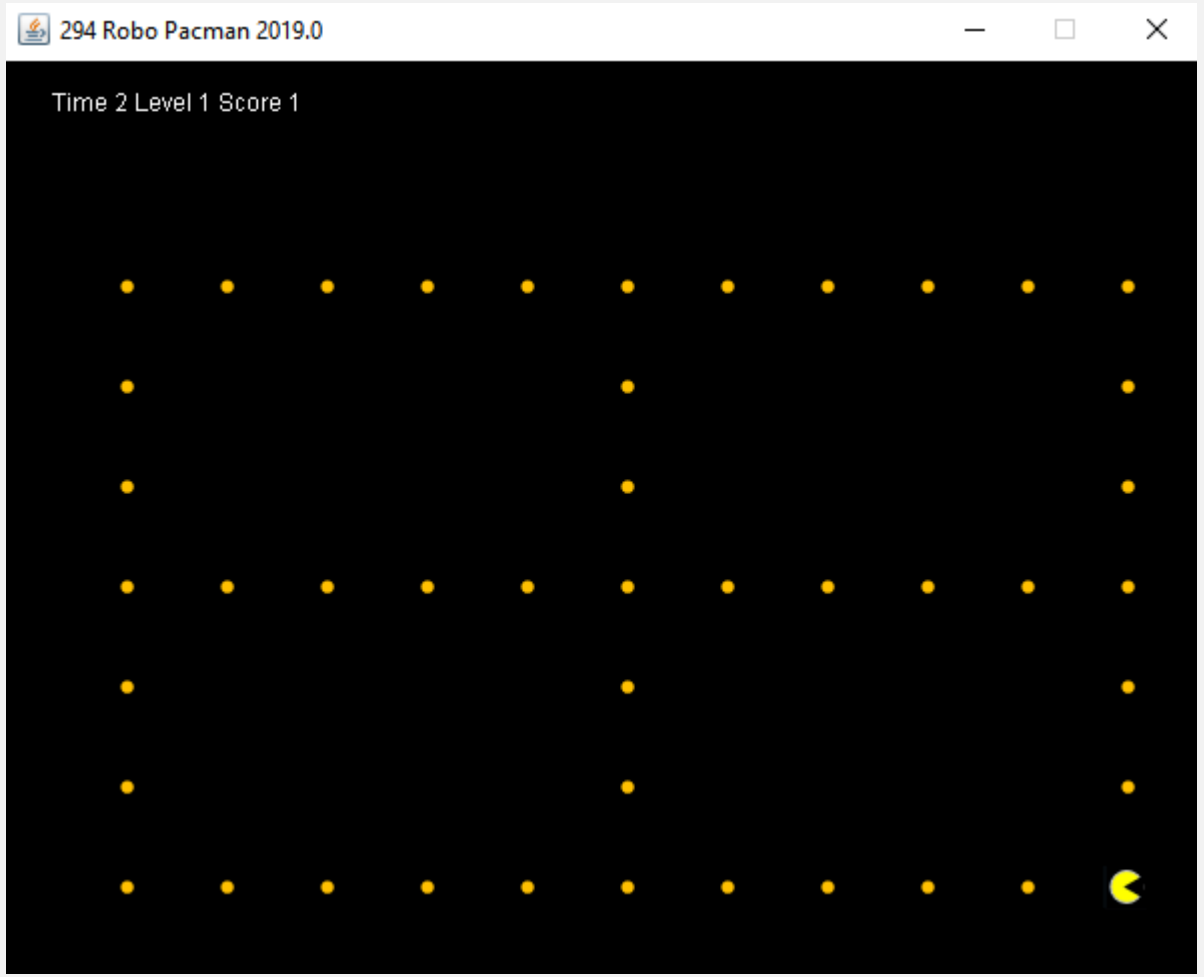
# RUN ROBO PACMAN

Run pacman.robot.Robot



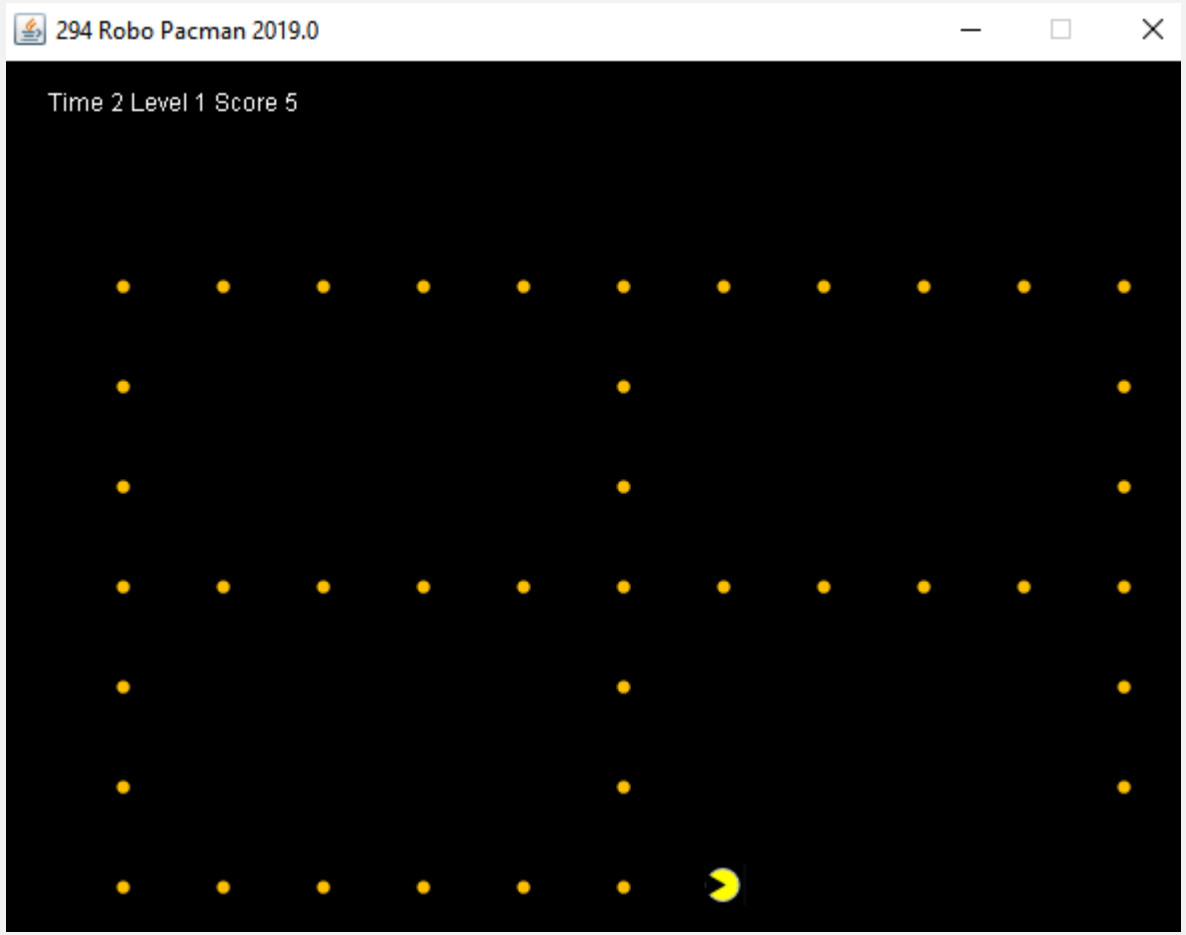
# CHALLENGE #1 – SPIN ONCE

- Make Pacman spin 360 degrees and then stop
- Create a new command called SpinOnce
  - Use pacman.commands.SpinForever as an example
- Change AutoGroup to call your new command
- Run Robot on level 1 to test



# CHALLENGE #2 – EAT THE DOTS

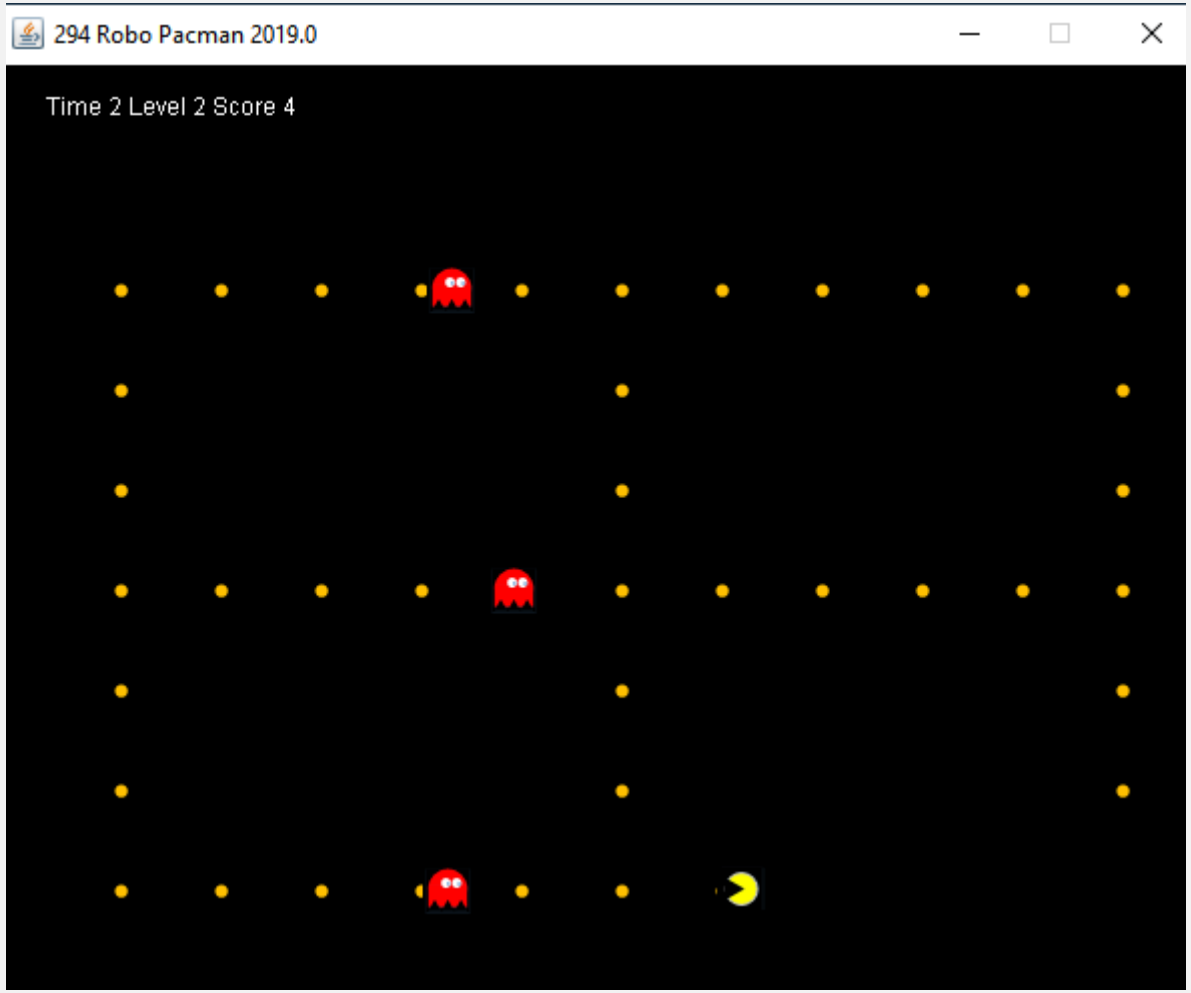
- Make Pacman move around the field and eat all the dots
- Create a new command called EatDots
  - Use pacman.commands.SpinForever as an example
- Change AutoGroup to call your new command
- Run Robot on level 1 to test



# CHALLENGE #3 – AVOID THE GHOSTS

- Make Pacman move around the field and eat all the dots while avoiding the ghosts

- Run Robot on level 2 to test



# GHOST SENSOR

Use the directional radar to detect if a ghost is directly in front of you

```
int ping = Robot.ghostSensor.getDirectionalRadar();
if (ping > 0) { // run! }
```

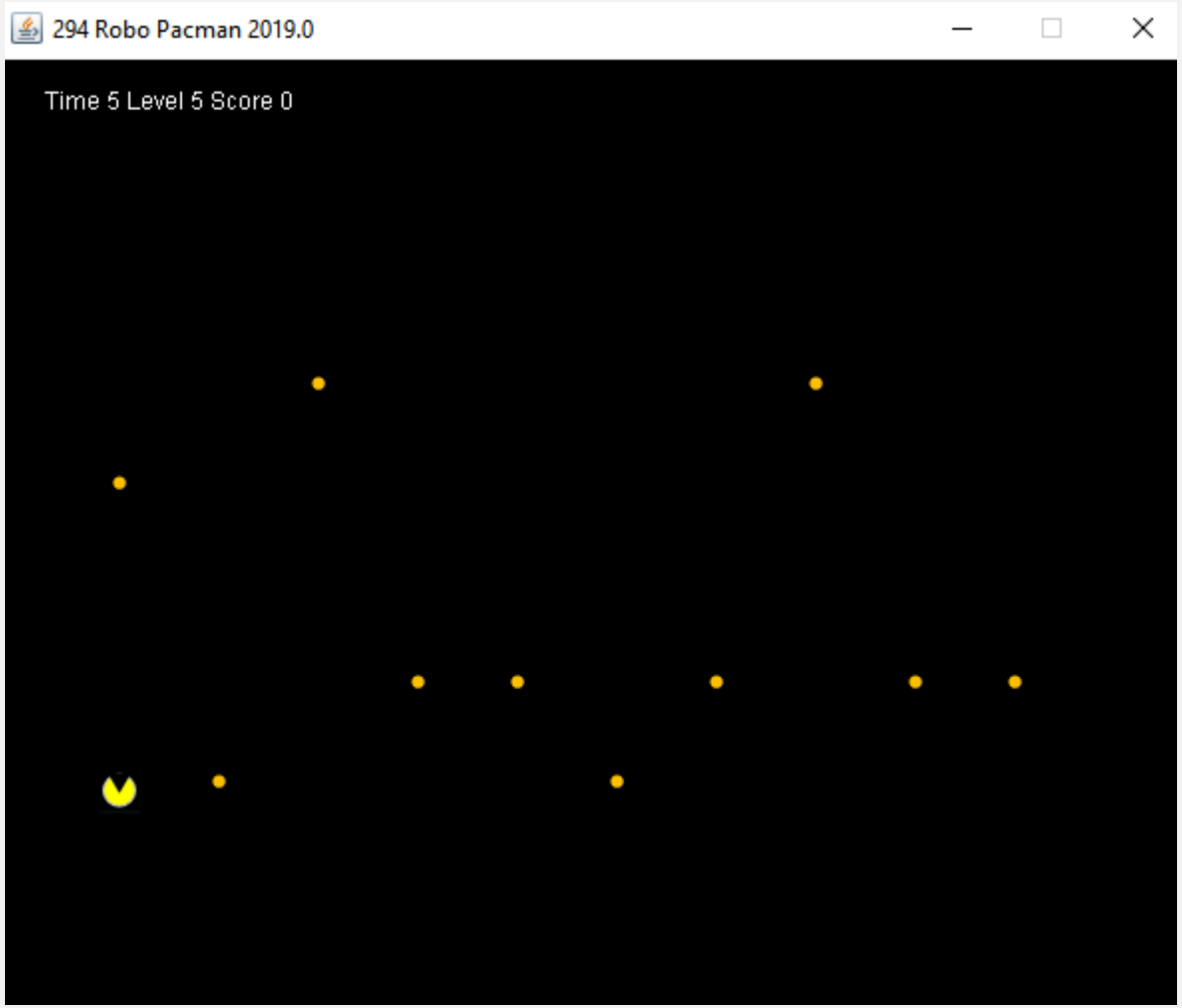
If you want to check for ghosts in all directions use the non-directional version

```
int ping = Robot.ghostSensor.getRadar();
```

The number returned from the sensor represents how many moves away the ghost is

# CHALLENGE #4 – FIND THE DOTS

- Make Pacman find all the randomly placed dots
- Use the DotSensor to find a path to the next dot
- Run Robot on level 5 to test



# DOT SENSOR

The dot sensor will give you the coordinates of all the dots

```
public int[][] getDotLocations()
```

# CHALLENGE #5 – FIND THE DOTS AND AVOID THE GHOSTS

- Make Pacman find all the randomly placed dots but avoid the ghosts
- You will need a path finding algorithm
- Run Robot on level 6 to test