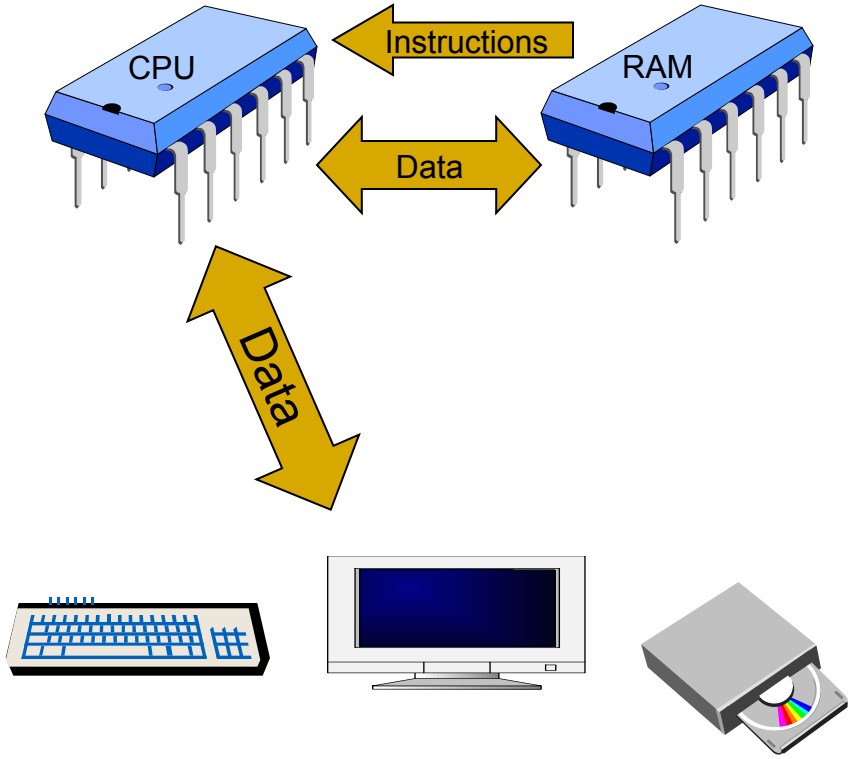
Introduction to Computer Programming

Theodore Norvell,
Dennis Peters, and
Lori Hogan
Enrichment Program, Memorial University
2004--2012

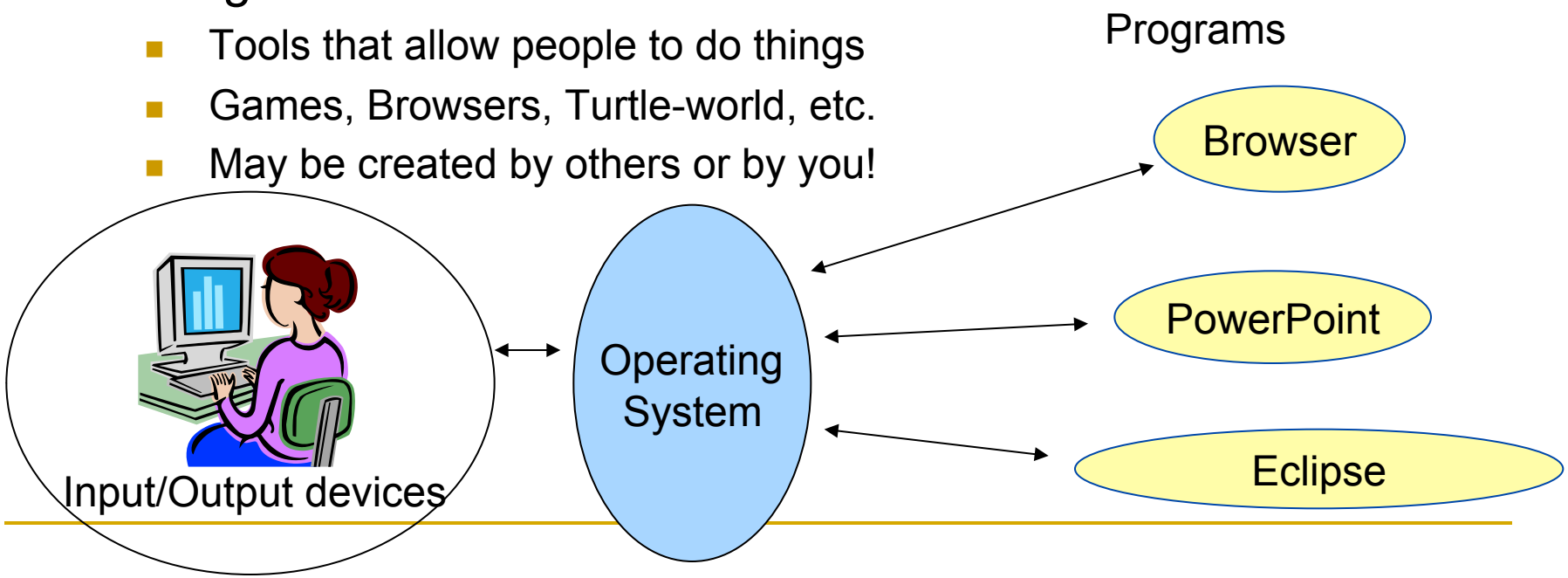
What is a Computer?

- Hardware view
  - Processor
  - Memory (RAM)
    - Containing
      - Data
      - Instructions
  - Input/output devices
    - Monitor
    - Keyboard
    - Mouse
    - Hard drive
    - Floppy drive
    - CD drive
    - Etc



What is a Computer?

- Software view
  - Operating system
    - Intermediary between the user, the hardware resources, and the various application programs
  - Programs
    - Tools that allow people to do things
    - Games, Browsers, Turtle-world, etc.
    - May be created by others or by you!



Writing your own program

- A **program** is a recipe for action.
  - It tells the computer how to act in response to each possible input.
- Programs may be written in the language the computer understands (machine language) or in a "high-level programming language" such as Java or C++.
- In this course we will modify a program written in the high-level **Java programming language**.
- Computers don't "understand" Java, but Java can be translated to machine language by another program.

Why program?

- No existing program to do what you want
- It's a part of your job
- To learn about computation
- For fun and personal satisfaction

Who programs

- Some professions demand software engineering skills as a central skill
  - Professional Software Engineers
  - Computer Scientists
  - Computer Engineers
- But in many other professions software engineering skills can be a useful
  - Other Engineers and other mathematical scientists
  - Physical Scientists
  - Artists

What is Programming?

- Telling the computer what you want it to do.
- Instructions are written in a programming language (e.g., Java, C++).
- An important part of both Computer Engineering and Computer Science.

Why programming is challenging

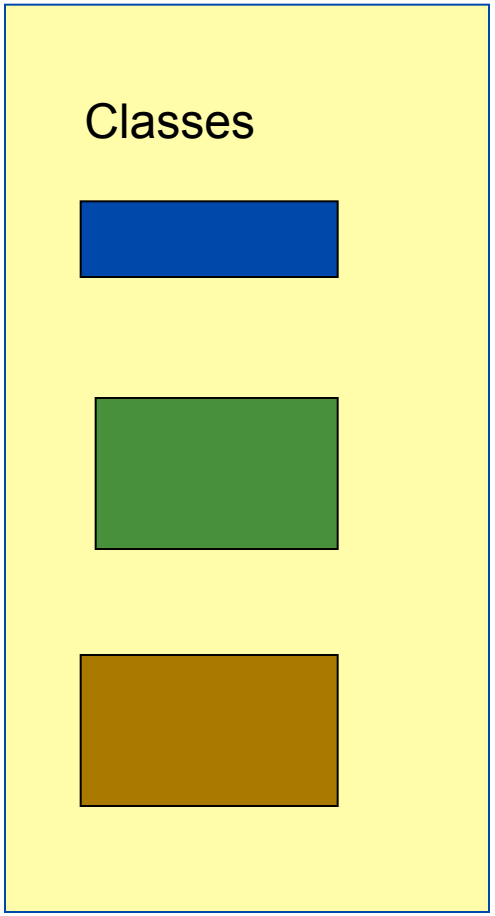
- Programming involves logic and time
  - It requires *imagination!*
  - It requires precise *reasoning!*
  - A program can be seen as a huge mathematical formula
  - Multiple activities interact in complex ways

- Most programs are the work of many people
  - This means that good communication skills, clarity about what you are doing and what you plan to do, and teamwork are often required.
  - Understanding both programming and the problem are important.

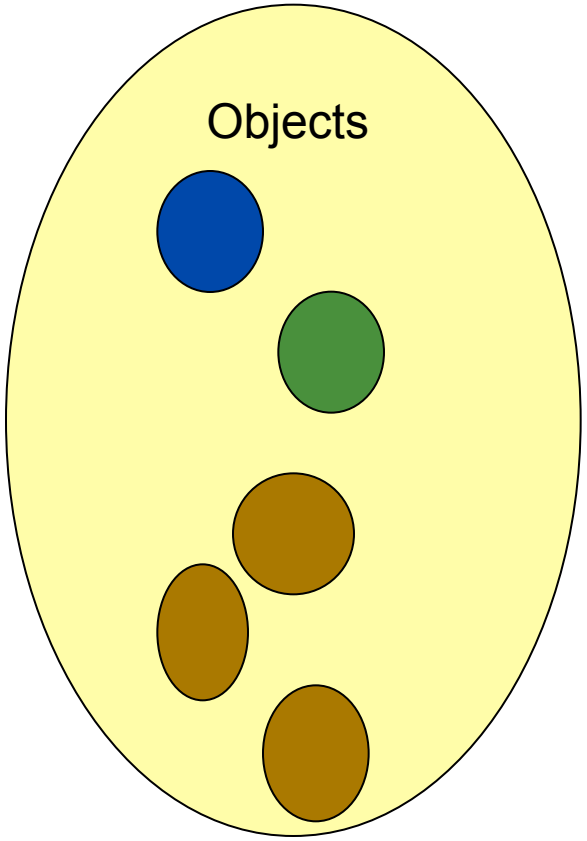
Java

- Java is a high-level programming language
  - A language for representing instructions to the computer
- Each Java program is written as a set of *classes*
- Each class describes the behaviour of zero, one, or more *objects*
- When the program is run (executed) it consists of one or more *objects* as described by the *classes.*

Java — Classes and Objects



A Java program



An execution of the program

What are objects?

- Represent things or concepts relevant to the problem and/or its solution.
  - Real-world things: car, person, apple.
  - Concepts: time.
  - Program things: button, window.
- Each object is an *instance* of a *class* that defines its behaviour.
- Each object is stored in memory at some location.
- An object may have a name (or more than one name).

Object-oriented Programming

- Design programs in terms of the concepts or things (*objects*) that are relevant to the task at hand.
- Objects interact by sending messages to each other. When an object receives a message, it:
  - performs some action, or
  - provides some information to the sender.
- Objects that behave the same are said to be in the same *class*.
- Object Oriented Programming (OOP): define classes and their behaviour.

Turtle World

- In this course you will program by modifying a program called “Turtle World”
- (Remember programs are usually the work of many people)

Classes in the Turtle-World

- Turtle — represents a simulated robotic turtle.
- Arena — the part of the screen the Turtles are displayed in.
- Log — a part of the screen to which you can send text.
- TurtleController — a class that describes how the program should react to user input.

Some Commands that Send Messages

Basic commands:

- send a message to an object.
  - Example:

```
crush.setPosition( 50, 20 ) ;
```

  - Sends a request to an object named “crush” to change its position to (50, 20).

Some Commands that Send Messages

Exercise:

- Find the “TurtleController” class.
- Find the “start” method. It looks like this

```
public void start() {
}
```

- Add the “message send command” crush.setPosition(50, 20) ; so that it looks like this.

```
public void start() {
    crush.setPosition( 50, 20 ) ;
}
```

- Save: Save the TurtleController class..
- Run: click on the “Run” on the “Run” menu.
- Try clicking on the “start” button. What happens.

Some Commands that Send Messages

Some messages result in an answer.

- send a message to an object and record the answer
  - Example:

```
double x = crush.getPositionX() ;
```

  - sends a “getPositionX” message to the turtle and names the answer “x”.

Some Commands that Send Messages

Exercise

- Change the “start” method to look like this

```
public void start() {
   crush.setPosition( 20, 50) ;
   double x = crush.getPositionX() ;
   double y = crush.getPositionY() ;
   log.println(x) ;
   log.println(y) ;
}
```

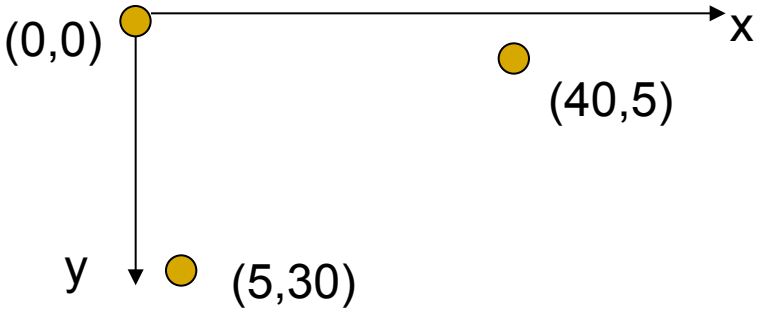
- The command “**double** x = crush.getPositionX() ;” sends a “getPositionX” message to the turtle, naming the answer “x”.
- Try running Turtleworld now.

Things to Know in Turtle-World

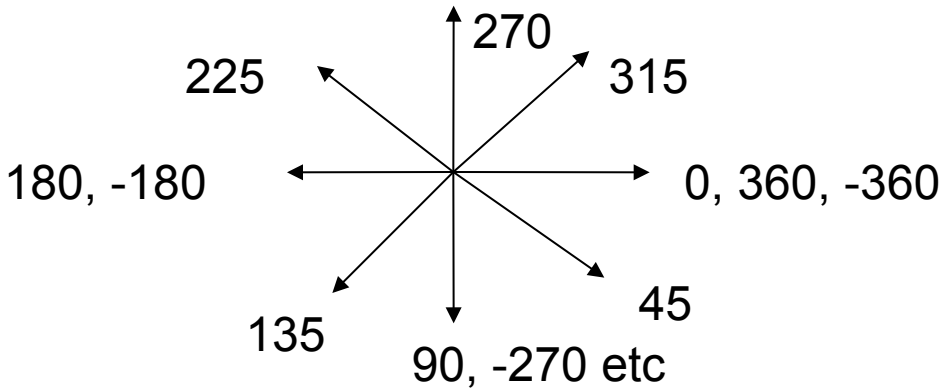
- What is a pixel?
  - It is the smallest possible block on the computer's screen.
- What do we mean by "speed"?
  - We mean how fast the turtle is going in pixels per second.
- What is "rate of change of speed"?
  - Similar to acceleration, it means how fast the turtle is speeding up, in pixels per second per second.
- What is "spin"?
  - Spin is how fast the turtle is spinning in degrees per second.

Things to Know in Turtle-World

- What is meant by position?
  - The position of a turtle is the location of the turtle in comparison to the top left-hand corner



- What is meant by orientation?
  - The orientation of the turtle is the way it is facing compared to the right side of the arena. Like a compass, the measurement goes clock-wise



Sending messages to the turtle

- Messages (see companion document for full list).
  - crush.setSpeed(s) ;
- Exercise: make the turtle start
  - Remove the commands you earlier added to “start” so that it looks like this again

```
public void start() {


}
```

  - Change the “start” method so that it tells the turtle to move at a speed of 50 pixels per second.

Sending messages to the turtle (cont.)

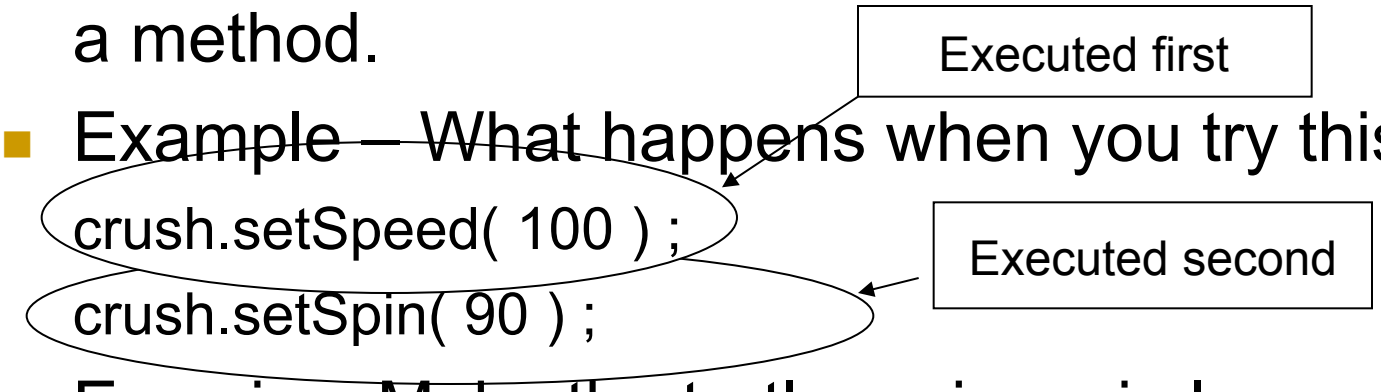
- Save: Save the TurtleController class..
- Run: click on the “Run” on the “Run” menu.
- Try clicking on the “start” button.
- What happens?
- Try clicking on the “stop” button.
- What happens?
- Close the Turtle World window.

Sending messages to the turtle (cont.)

- Making the turtle stop
  - Find the “method” that the TurtleController executes in reaction to to the “stop” button being clicked.
  - Change this “method” to make the turtle stop.
- Making the turtle go slower/faster or spin at start
  - How would you change the “start method” so that the turtle goes slower or faster? So that the turtle spins?

Sequences of commands

- Commands are followed in the order they appear in a method.
- Example – What happens when you try this?



- Exercise: Make the turtle go in a circle.
  - Modify the “circle” method to make the turtle travel in a circle.
- Exercise: Change around the order that commands are given
  - Does the order of the commands matter here? In general?

Sequences of commands

- The messages to the turtle
  - crush.penDown() ;
  - crush.penUp() ;

  control whether the turtle leaves a trace behind it of where it has been.
- Exercise: Modify the “circle” method to draw a circle.
- Exercise: How would you modify the “circle” method to only draw part of a circle?
- Exercise: Modify the “mySequence” method to do four different commands (see the companion document for ideas) and then change the order of them to see if it makes a difference.

Getting information from the turtle

- The commands
  - **double** x = crush.getPositionX() ;
  - **double** y = crush.getPositionY() ;
  - **double** s = crush.getSpeed() ;
  - **double** alpha = crush.getOrientation() ;
  - **double** vx = crush.getVelocityX() ;
  - **double** vy = crush.getVelocityY() ;
  - **double** a = crush.rateOfChangeOfSpeed() ;
  - **double** omega = crush.spin() ;

  obtain information about the turtle and give that value a name. ("double" is short for "double precision number", i.e. these names stand for numbers.)

Getting information from the turtle

- Exercise: Turning left.
  - Modify the “left” method to turn the turtle left by 30 degrees.
  - Hint: first give a name to the current orientation, then use a “setOrientation” message to turn the turtle. Orientations are in degrees.
  - Another hint. You can add two numbers using the + operator. For example: X+Y where X and Y are either numbers or names that stand for numbers.
  - Question: Does the order of the commands matter?
- Exercise: Turning right.
  - Do the same as the exercise above for the “right” method

Making choices

- Choose a sequence of commands depending on whether a condition is true or not

  if (*condition* **) {**

  a sequence of commands

  } else {

  another sequence of commands

  }

- Many real-life examples of making choices, can you think of any?

Making choices

- Example: Speed up the turtle, but not faster than 100

```
public void faster() {
    double s = crush.getSpeed() ;
    if ( s + 5 > 100 ) {
       crush.setSpeed( 100 ) ;
    } else {
       crush.setSpeed( s + 5 ) ;
    }
}
```

- Exercise: Modify “slower” so that the speed is not set to less than 0.
- Exercise: Modify “mySequence” so that if the turtle is facing left it turns to the right and vice-versa.

Pausing

- The basic command

  pause(*t*) ;

  causes the TurtleController to wait for *t* seconds before executing the next command.
- Exercise: Modify the “start” method so that when the button is pressed, the turtle travels for 5 seconds and then stops (we will change it back later!).
- Exercise: Draw a square.
  - Modify the “square” method so that the turtle draws a square by traveling right for 1 second, turning right, traveling down for 1 second, turning right, traveling left for 1 second, turning right, traveling up for one second and then stopping.

Repetition: Do it some number of times.

- If some commands should be repeated a fixed number of times:

```
for (int i = 0; i < N; i++) {
    sequence of commands
}
```

- Can you think of some real-world examples of where an iteration loop with a given number of repeats (also called a “for” loop) is used?

"For" loop example

- Example: Another way to draw a square.

```
public void hexagon() {
  int N = 4 ;
  crush.penDown() ;
  crush.setSpeed(100) ;
  for (int i = 0 ; i < N ; i++) {
    pause(1) ;
    double alpha = crush.getOrientation() ;
    crush.setOrientation( alpha + 90 ) ;
  }
  crush.setSpeed(0) ;
}
```

int stands for "integer number"

Command sequence repeated 4 times.

“For” loop exercises.

- Modify “hexagon” to create an 6 sided polygon.
- Modify your “sequence” method
  - Put a “for” loop around your sequence and make it go for five iterations.
  - Add the command

    crush.penDown() ;

    right before it.
  - What types of drawings do you come up with?

Repetition: Repeat while needed

- Repeat something as long as some condition is true

  while (*condition* **) {**

  sequence of commands

  }

- What real-life examples can you come up with where something is repeated as long as something else is true (also called a “while” loop because while something is true we do this)?

"While loop" example

- Make a new button with this method:

```
public void changeColor() {
    while( crush.getSpeed() > 0 ) {
        crush.setColor(Color.blue) ;
        pause(0.5) ;
        crush.setColor(Color.red) ;
        pause(0.5) ;
        crush.setColor(Color.black) ;
        pause(0.5) ;
    }
}
```

- Try starting the turtle. Then click changeColor. When does the turtle stop changing color.

Combining Commands

There are a number of ways to combine simpler commands to make more complex commands

- Sequential — do one thing after another.
  - In Java, put one command after another.
- Choice — do one command or another depending on a condition
  - In Java, use `if-else`
- Repetition — do the same things more than once
  - In Java, use `for`-loops or `while`-loops

"While" loop exercises

- Change the "for" loop in your "sequence" method to a "while" loop that keeps going until the speed is zero (you may have to use "slower" while your turtle is running through the sequence to slow it down).

- Change this again so that your sequence keeps going until the speed is either over 100 or under zero (again, you may have to use the "faster" and "slower" buttons).

“While” loop exercises (continued)

- How would you keep your sequence of commands going “forever”, until you hit “stop”?

New methods

- A “method” is a sequence of commands with a name.
- Define a new method in a class

```
public void name ()
{
   sequence of commands
}
```

- The method name becomes a new message that the objects of the class understand.

New Method Exercise

- Create new method octagon
  - Hint: copy, paste and modify the "body" of the hexagon method.
- Add octagon to the list of methods activated by buttons
  - Add the word "octagon" to the list named "methodNames" near the start of the TurtleController.java file.

Parameters

- Parameters allow variations on a single method.
- Example: When we use the "pause" command we put a number in the brackets that tells the method how long to pause for. it looks like

- t is a parameter. It is a name that stands for different values at different times. E.g. in the command

```
pause( 0.5 ) ;
```

t stands for 0.5.

Parameters exercise

- Try creating a method named “polygon” with a parameter N.
- Then modify square, hexagon, and octagon to use it.
- Parameters and methods allow common sequences of commands to be programmed once and used over again.
  - This is called “procedural abstraction”.

More objects

- Declare that our system contains a new object

  ```
  private ClassName objectName = new ClassName(…);
  ```

- Example:
  - **private** Turtle squirt = **new** Turtle(Color.RED) ;
- Exercise: add a new turtle (call it what you like – I called mine "squirt")
  - Find the declaration of "crush" in the TurtleController class.
  - Add a declaration for "squirt" (as above)
  - Find where crush is added to the arena
  - add a command to add the "squirt" to the "arena".
  - Add some buttons to control "squirt"

What is Programming?

- An Art or Craft?
  - Is writing a program like writing a book?
  - Is it all about effective communication?
- A Mathematical Science?
  - Logic is the mathematics of relationships.
  - Programming is the mathematics of relationships evolving through time.
- Engineering?
  - The analysis and design of artifacts.
  - Programs are artifacts that must be designed and may be analyzed.
- Perhaps it is all three.

Interesting problems in software engineering.

- How to solve problems with minimum execution time
- How to solve problems with minimum space
- How to get a large number of people to cooperate effectively to create large programs
- How to specify what a program should do
- How to avoid programming mistakes (bugs)
- How to find any remaining programming mistakes
- How to know whether a program does what it should

The end