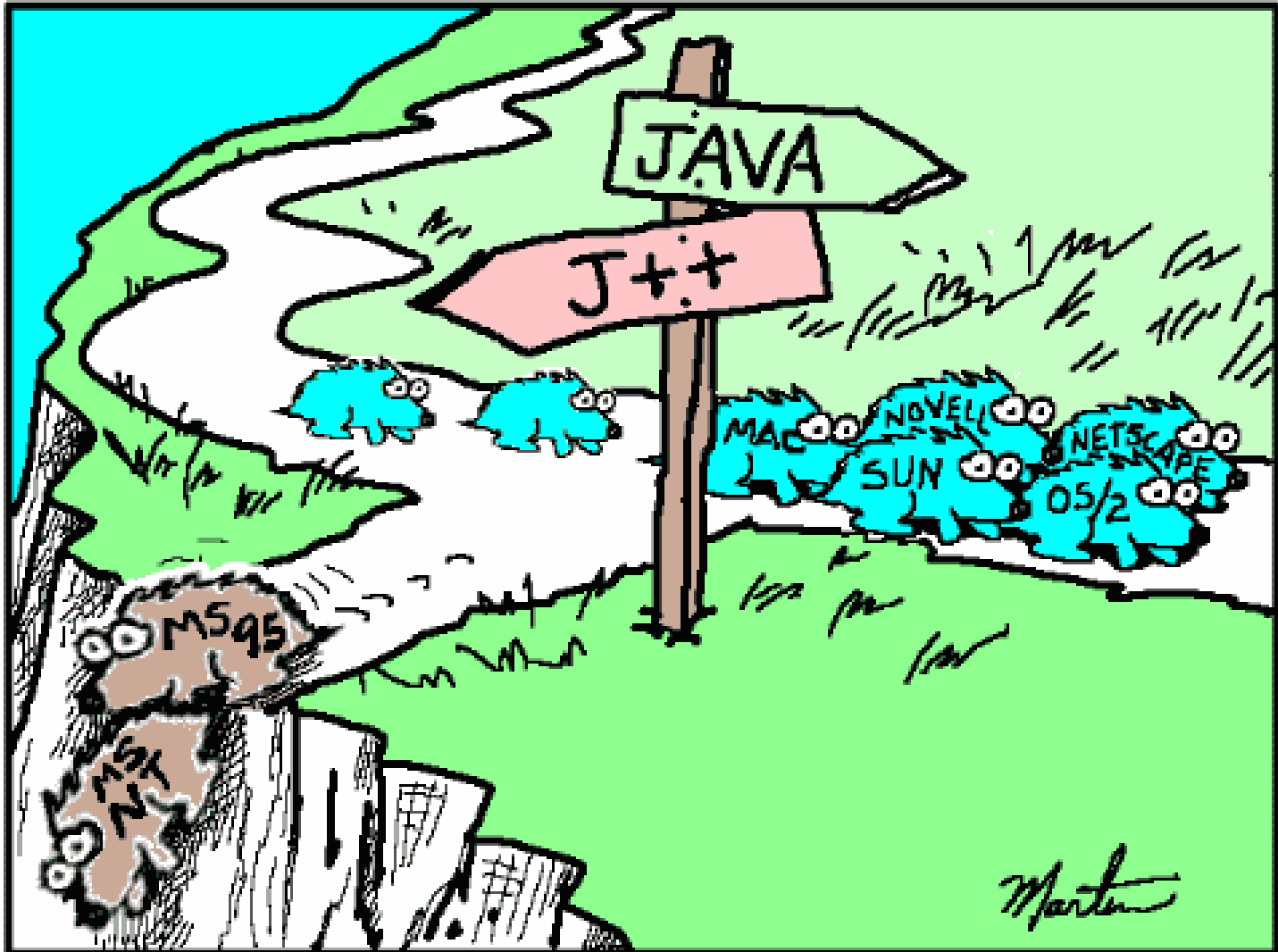


Loopy



while loop with sentinel control

```
Average2.java *
1 import java.text.DecimalFormat;
2
3 // Java extension packages
4 import javax.swing.JOptionPane;
5
6 public class Average2 {
7
8     // main method begins execution of Java application
9     public static void main( String args[] )
10    {
11        int gradeCounter, // number of grades entered
12            gradeValue, // grade value
13            total; // sum of grades
14        double average; // average of all grades
15        String input; // grade typed by user
16
17        // Initialization phase
18        total = 0; // clear total
19        gradeCounter = 0; // prepare to loop
20
21        // Processing phase
22        // prompt for input and read grade from user
23        input = JOptionPane.showInputDialog(
24            "Enter Integer Grade, -1 to Quit:" );
25
26        // convert grade from a String to an integer
27        gradeValue = Integer.parseInt( input );
28
```

while loop with sentinel control continued...

```
29     while ( gradeValue != -1 ) {
30         // add gradeValue to total
31         total = total + gradeValue;
32         // add 1 to gradeCounter
33         gradeCounter = gradeCounter + 1;
34         // prompt for input and read grade from user
35         input = JOptionPane.showInputDialog(
36             "Enter Integer Grade, -1 to Quit:" );
37         // convert grade from a String to an integer
38         gradeValue = Integer.parseInt( input );
39     }
40
41     // Termination phase
42     DecimalFormat twoDigits = new DecimalFormat( "0.00" );
43
44     if ( gradeCounter != 0 )
45     {
46         average = (double) total / gradeCounter;
47
48         // display average of exam grades
49         JOptionPane.showMessageDialog( null,
50             "Class average is " + twoDigits.format( average ),
51             "Class Average", JOptionPane.INFORMATION_MESSAGE );
52     }
53     else
54         JOptionPane.showMessageDialog( null,
55             "No grades were entered", "Class Average",
56             JOptionPane.INFORMATION_MESSAGE );
57
58     System.exit( 0 );    // terminate application
59
60 } // end method main
61
62 } // end class Average2
```

Stepwise Refinement

When attempting a complex problem such as that posed in the login/password program begin by breaking the program into smaller steps. Compile, test and debug these smaller sections before adding additional complexity to the program.

The following 3 steps are just a suggestion as to how you might go about breaking a larger program into smaller sections.

1. Have user enter a login and use if statement to determine if correct login was entered.
2. Use a nested loop to determine if password is correct after user has entered correct login.
3. Use a while loop to track the number of incorrect entries and close the program after a certain number of failed attempts.

Login and Password

Start Small....code this amount and test with correct and incorrect logins

```
LoginPassword.java
1 import javax.swing.JOptionPane;
2 public class LoginPassword
3 {
4     public static void main(String args[])
5     {
6         String login = "toast";
7         String password = "butter";
8         String loginMatch;
9         String passwordMatch;
10
11         loginMatch =
12             JOptionPane.showInputDialog("Please enter your login");
13         passwordMatch =
14             JOptionPane.showInputDialog("Please enter your password");
15
16         if(loginMatch.equals(login))
17         {
18             JOptionPane.showMessageDialog(null,"Correct login,Welcome.");
19         }
20         else
21         {
22             JOptionPane.showMessageDialog(null,"Incorrect login, please enter correct login.");
23         }
24     }
25 }
```

Add a little more complexity

Nested if statement to reflect a correct login AND password

```
15
16     if(loginMatch.equals(login))
17     {
18         if(passwordMatch.equals(password))
19         {
20             JOptionPane.showMessageDialog(null,"Welcome");
21         }
22         else
23         {
24             JOptionPane.showMessageDialog(null,"Incorrect password");
25         }
26     }
27     else
28     {
29         JOptionPane.showMessageDialog(null,"Incorrect login, please enter correct login.");
30     }
31 }
32 }
```

Correct login but incorrect password

Incorrect login

Test with every combination of correct and incorrect password/login combinations to ensure you are getting the expected results.

Limit attempts using while loop

Encompass entire if statements block with a while loop that exits after 5 failed attempts.

Declare attempts variable to track failed attempts

```
15  int attempts = 0;
16  while(attempts < 5)
17  {
18      if(loginMatch.equals(login))
19      {
20          if(passwordMatch.equals(password))
21          {
22              JOptionPane.showMessageDialog(null,"welcome");
23              break;
24          }
25          else
26          {
27              JOptionPane.showMessageDialog(null,"Incorrect password");
28              attempts++;
29              passwordMatch = JOptionPane.showInputDialog("Please enter your password");
30          }
31      }
32      else
33      {
34          JOptionPane.showMessageDialog(null,"Incorrect login, please enter correct login.");
35          attempts++;
36          loginMatch = JOptionPane.showInputDialog("Please enter your login");
37      }
38  }
39  }
```

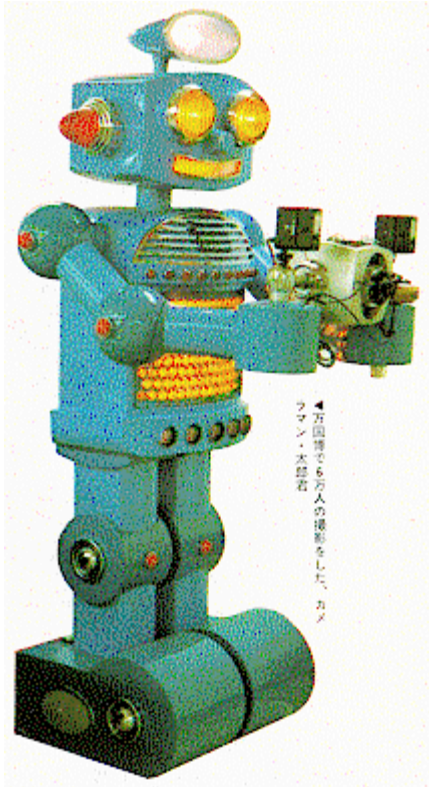
After successful login break out of while loop

Prompt for login after failed login attempt.

Failed login/password attempts result in attempts variable being incremented

Prompt for password after failed password attempt.

Robots

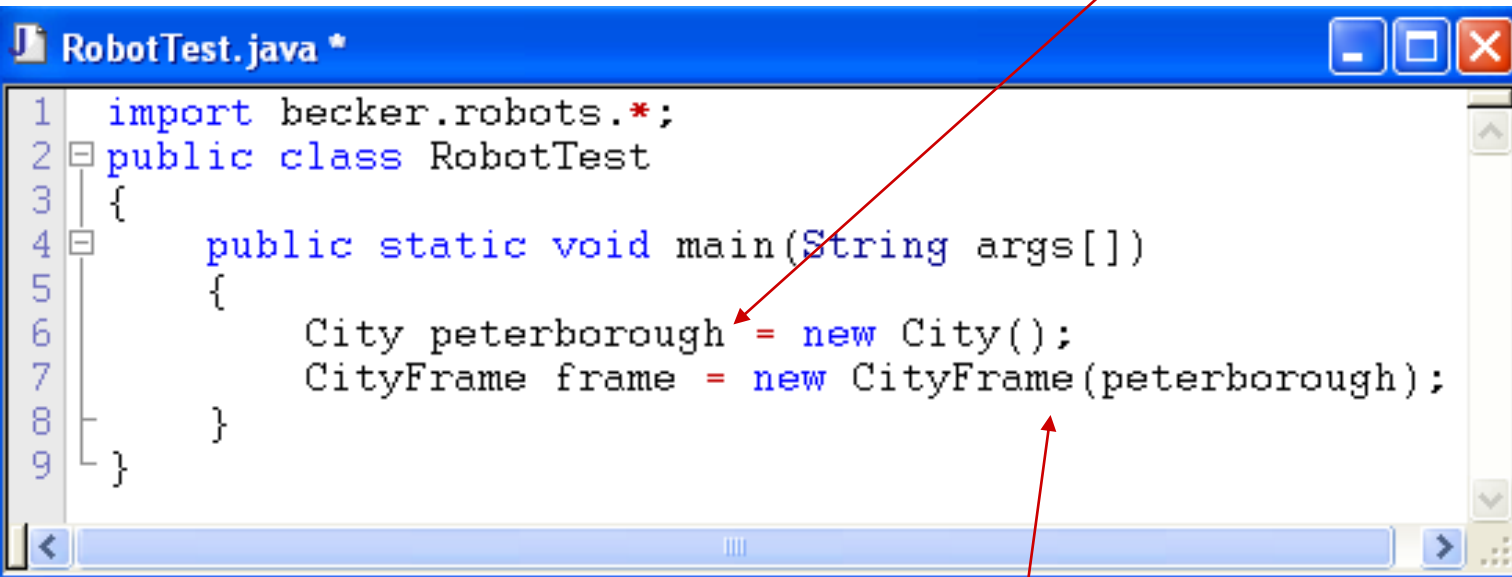


Make sure you have followed the directions in the previous lesson for downloading and installing the robot class packages (becker.jar)

Anatomy of a robot program

Here is the bare minimum code required to create a city and display it in a frame

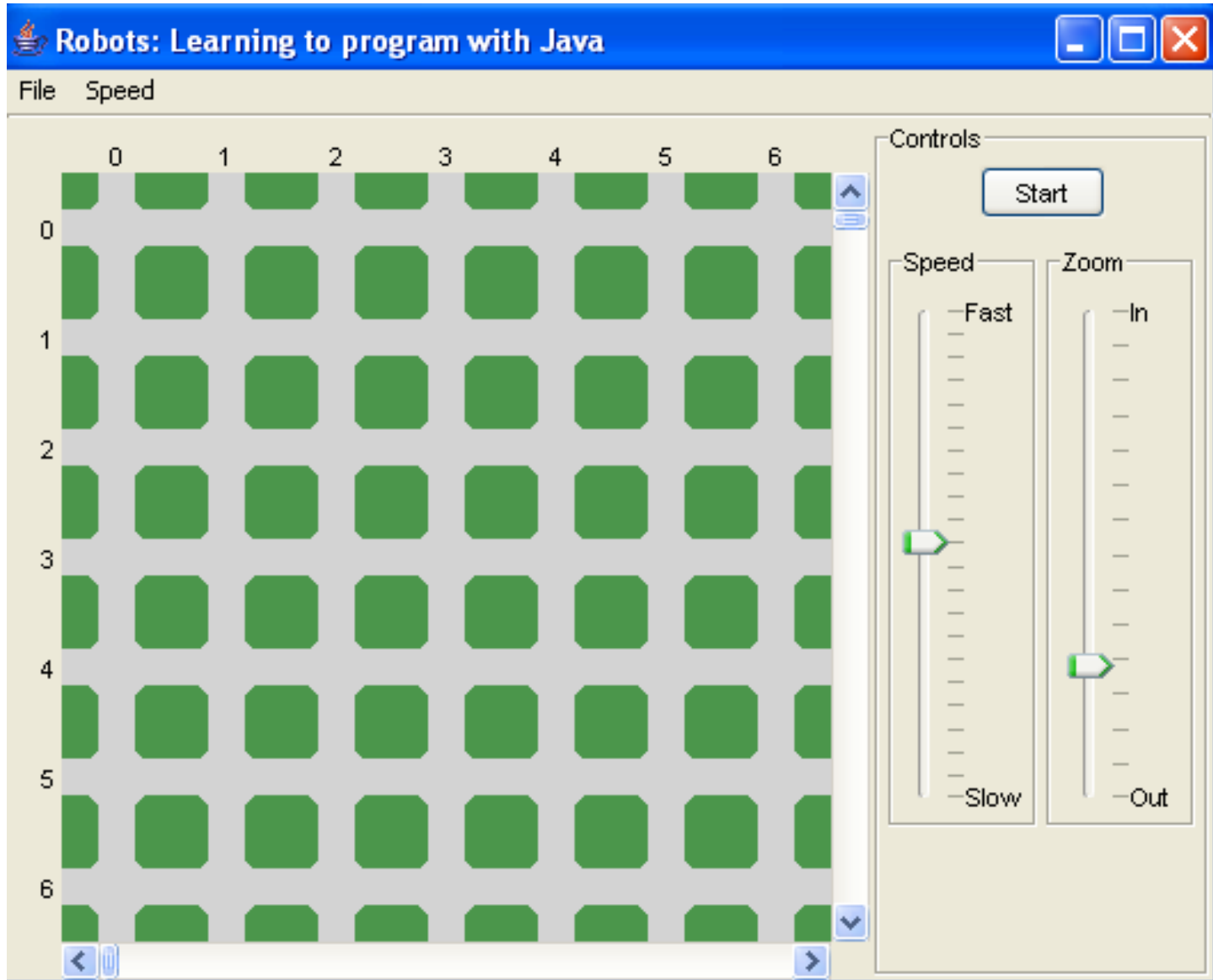
Declare city



```
1 import becker.robots.*;
2 public class RobotTest
3 {
4     public static void main(String args[])
5     {
6         City peterborough = new City();
7         CityFrame frame = new CityFrame(peterborough);
8     }
9 }
```

Embed city in a frame

The result...an empty city.



Lets add a robot...we'll call him bill

```
RobotTest.java *
1  import becker.robots.*;
2  public class RobotTest
3  {
4      public static void main(String args[])
5      {
6          City peterborough = new City();
7          CityFrame frame = new CityFrame(peterborough);
8
9          Robot bill = new Robot(peterborough, 2,4,Directions.SOUTH)
10     }
11 }
```

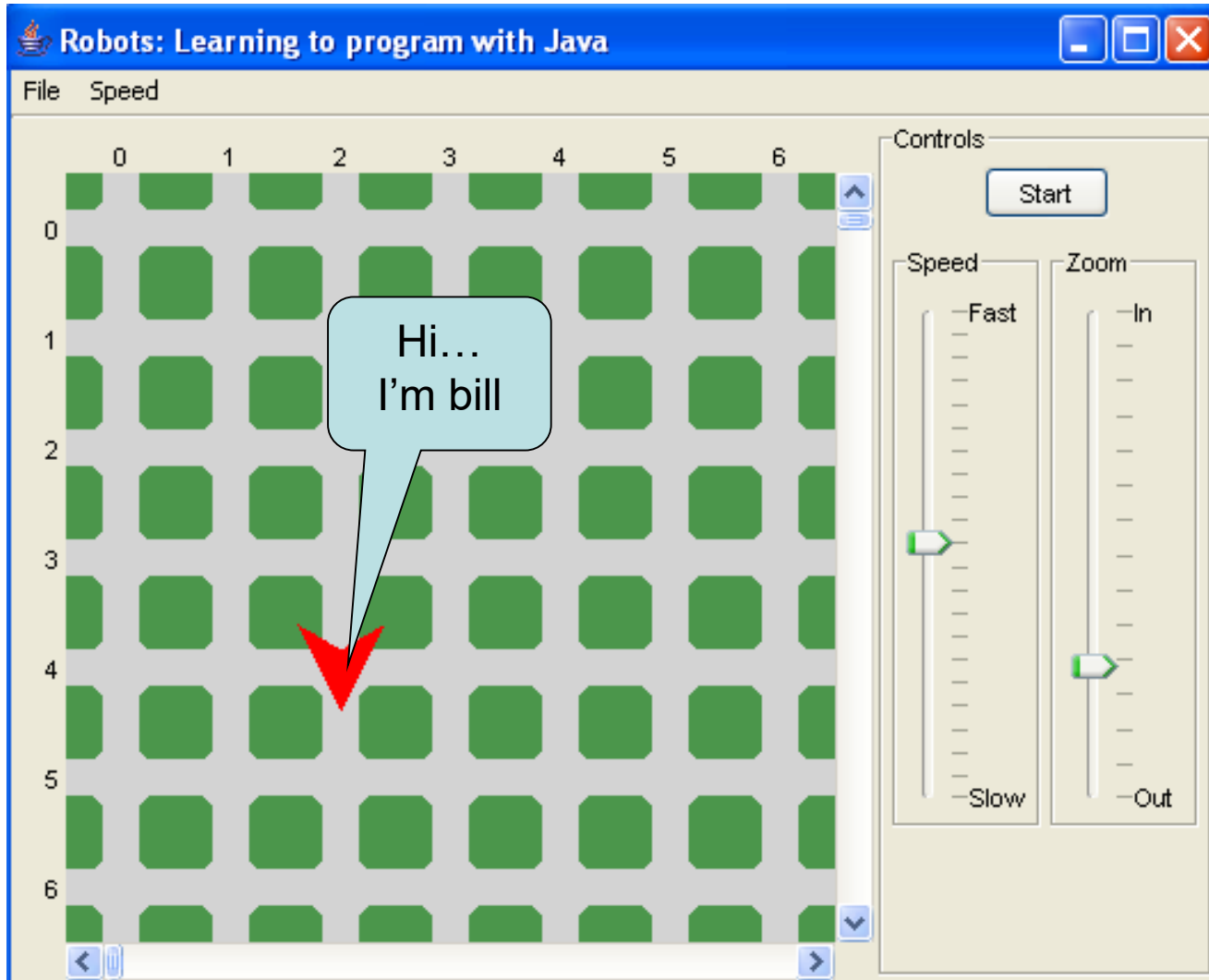
Parameter 1:
city where
robot exists

Parameter 2:
x axis

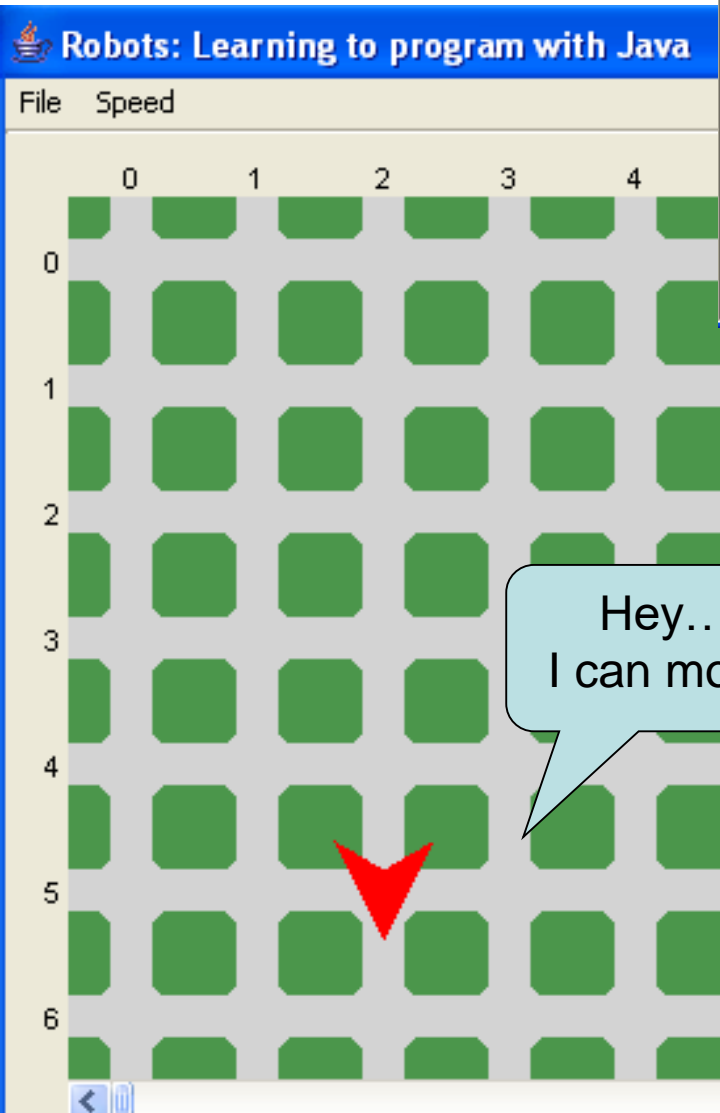
Parameter 3:
y axis

Parameter 4:
Direction robot
is facing

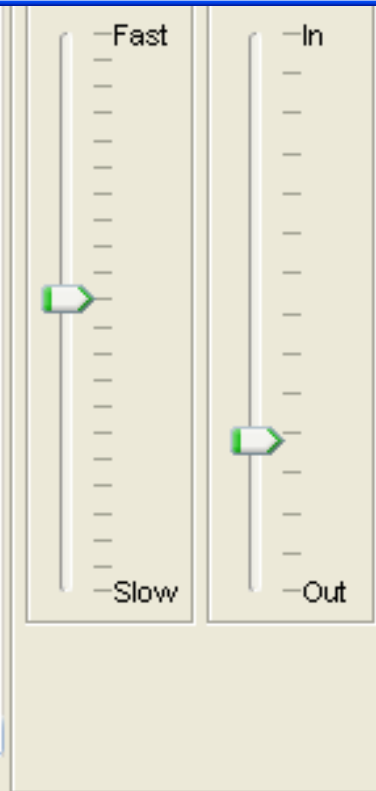
Meet bill....resident of Peterborough



See bill move...



```
RobotTest.java
1 import becker.robots.*;
2 public class RobotTest
3 {
4     public static void main(String args[])
5     {
6         City peterborough = new City();
7         CityFrame frame = new CityFrame(peterborough);
8
9         Robot bill = new Robot(peterborough, 2,4,Directions.SOUTH);
10
11         bill.move();
12     }
13 }
```



See bill turn....

The image shows a Java IDE window titled "RobotTest.java" with the following code:

```
1 import becker.robots.*;
2 public class RobotTest
3 {
4     public static void main(String args[])
5     {
6         City peterborough = new City();
7         CityFrame frame = new CityFrame(peterborough);
8
9         Robot bill = new Robot(peterborough, 2,4,Directions.SOUTH);
10
11         bill.move();
12         bill.turnLeft(); //3 left turns to turn right
13         bill.turnLeft();
14         bill.turnLeft();
15     }
16 }
```

The IDE also shows a window titled "Robots: Learning to program with Java" containing a grid-based simulation. The grid has columns labeled 0-4 and rows labeled 0-6. A red arrow points to the intersection of column 2 and row 5. A speech bubble next to the arrow says "Watch me now!!!". To the right of the grid is a control panel with two vertical sliders labeled "Slow" and "Out", each with a green arrowhead pointing to the right.

Where is bill?

Robots: Learning to program with Java

File Speed

0 1 2 3 4 5 6

0

1

2

3

4

5

6

Controls

Stop

Speed Fast Zoom -In

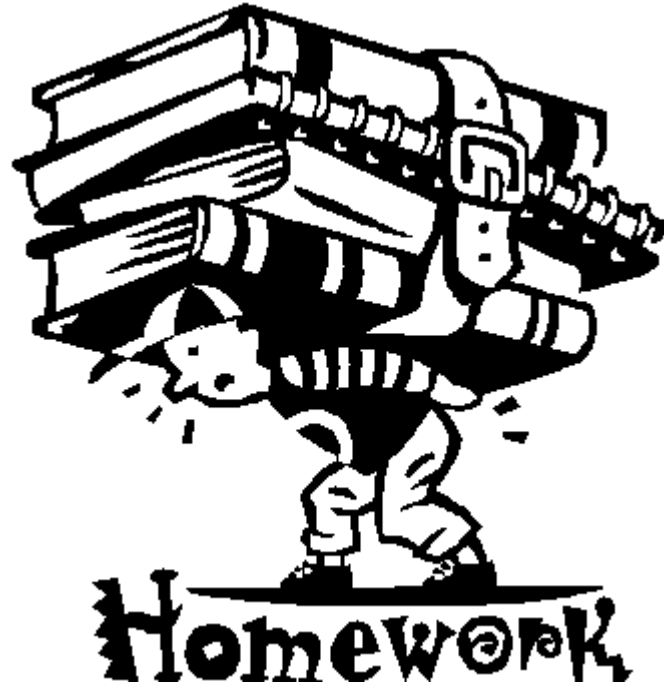
Message

Look for me on the corner of 5 and 5

OK

RobotTest.java

```
1 import javax.swing.*;
2 import becker.robots.*;
3 public class RobotTest
4 {
5     public static void main(String args[])
6     {
7         City peterborough = new City();
8         CityFrame frame = new CityFrame(peterborough);
9
10        Robot bill = new Robot(peterborough, 2,4,Directions.SOUTH);
11
12        bill.move();
13        bill.turnLeft();
14        bill.move();
15        bill.move();
16        bill.move();
17        int y = bill.getStreet();
18        int x = bill.getAvenue();
19        JOptionPane.showMessageDialog(null, "Look for me on the corner of "
20        + x + " and " + y);
21    }
22 }
```



Copy the code in the next slide. Now experiment with moving bill around the city using loops and if statements.

Home Work Code

RobotTest2.java

```
1 import javax.swing.*;
2 import becker.robots.*;
3 public class RobotTest2
4 {
5     public static void main(String args[])
6     {
7         City peterborough = new City();
8         CityFrame frame = new CityFrame(peterborough);
9         Robot bill = new Robot(peterborough, 2,4,Directions.EAST);
10
11         int x = bill.getStreet();
12         int y = bill.getAvenue();
13         while(x < 7)
14         {
15             bill.move();
16             x = bill.getAvenue();
17         }
18         bill.turnLeft();
19         while(y > 1)
20         {
21             bill.move();
22             y = bill.getStreet();
23         }
24
25         JOptionPane.showMessageDialog(null, "Look for me on the corner of "
26 + x + " and " + y);
27     }
28 }
```