



TURTLES

How Do They Work



How to Compile and Run

- Put turtleClasses.jar in the same directory as CS11TurtleGraphics.java

```
javac -cp "turtleClasses.jar:." CS11TurtleGraphics.java
```

```
java -cp "turtleClasses.jar:." CS11TurtleGraphics
```

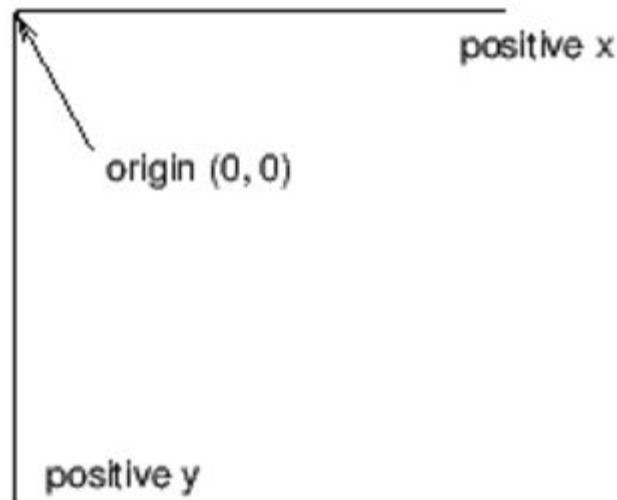
- This is because turtleClasses.jar is not part of the default Java classpath
- AKA when you try to use stuff that is in the turtleClasses.jar, it doesn't know where to look
- -cp modifies the classpath to include turtleClasses.jar

Graphics Coordinates

- Origin aka (0, 0) is in the TOP left corner of the screen
- x-axis increases going right
- y-axis increases going DOWN

Ex. (50, 125) is 100 pixels BELOW (50, 25)

Java graphical coordinates



The Turtle Library

- How do the Turtles work?
 - We make a World and a Turtle
 - We move the Turtle around
 - The Turtles leave lines behind
- Why are we using Turtles?
 - Helps programmers visualize what their code is doing by animating it
 - Introduction to object oriented programming
 - Basic introduction to using methods



Setting Up Your Turtle and World

- To create a World, make a World variable and call the World constructor
- World constructor takes in height and width

```
World yourWorld = new World(width, height);
```

- To create a Turtle make a new Turtle variable and call the Turtle constructor
- Turtle constructor takes in world and delay



```
CS11TurtleGraphics yourTurtle = new CS11TurtleGraphics(yourWorld,  
delay);
```

- To change color, size, visibility of Turtle look at the API

<http://cseweb.ucsd.edu/~ricko/CSE11/turtleClassesDocs/index.html?>

Example Code

```
public class CS11TurtleGraphics extends Turtle {  
    ...  
    // methods  
    public void drawLine(int x, int y) {...} // see slide 8  
    ...  
  
    public static void main(String[] args) {  
        World yourWorld = new World(500, 500);    // world is 500x500  
        CS11TurtleGraphics yourTurtle = new CS11TurtleGraphics(yourWorld, delay);  
        yourTurtle.drawLine(40, 40);    // draws line starting from (40, 40)  
    }  
    // DO NOT USE HARD CODED NUMBERS - SEE SLIDE 10  
}
```

Methods for Movement

- penUp(): when turtle moves, no line is drawn
- penDown(): when turtle moves, a line is drawn
- turnLeft(): turns turtle counter clockwise 90 degrees
- turnRight(): turns turtle clockwise 90 degrees
 - Ex. if turtle is facing south, turnLeft() will turn turtle east and turnRight() will turn turtle west
- turn(double degrees): turns turtle by a number of degrees
- forward(int pixels): moves turtle forwards pixels
- backward(int pixels): moves turtle backwards by pixels
- look at Turtle API for more movement options

[http://cseweb.ucsd.edu/~ricko/CSE11/turtleClassesDocs/index.html?
turtleClasses/SimpleTurtle.html](http://cseweb.ucsd.edu/~ricko/CSE11/turtleClassesDocs/index.html?turtleClasses/SimpleTurtle.html)

Example Code

```
private void drawLine(int x, int y) {  
    penUp();           // when turtle moves, no line is draw  
    moveTo(x, y);      // position turtle at starting point  
    turnToFace(getXPos(), getYPos()+1); // faces turtle down  
    penDown();         // when turtle moves, a line is drawn  
    forward();         /* moves turtle forward 100 pixels  
                        also draws a line 100 pixels long */  
}
```

Example vs Given



World



World



Using Constants aka Avoid “Magic Numbers”

- Notice that all numbers are saved as variables
- We used hard coded constants in Slide 6 for the sake of example
- Do not do this in your code: hard to modify, hard to understand
- Few situations where hard coded constants are ok
 - When incrementing or decrementing (+1 or -1)
 - When checking if a variable is zero (== 0)
 - When dividing or doubling something AND it is clear why you are doing so (*2 or /2)
- Otherwise, use of hard coded constants will result in LOSS OF POINTS

Question Time



Demo Time

