
Introduction to Computational Thinking

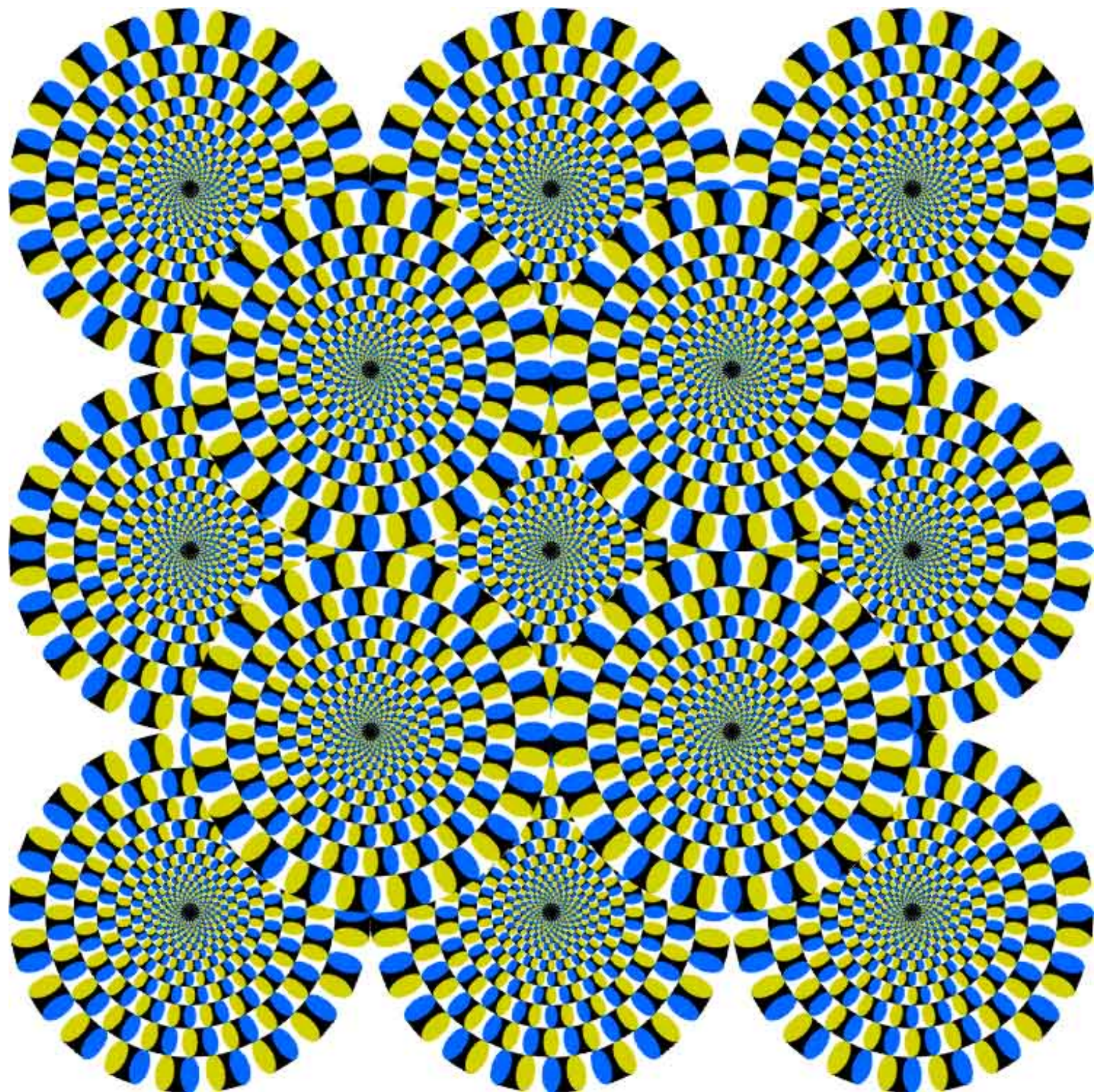
Lecture #5: *Java Graphics*

Qiao Xiang, Qingyu Song

<https://sngroup.org.cn/courses/ct-xmuf25/index.shtml>

10/29/2025

This deck of slides are heavily based on cs112 at Yale University and cs101 at UCAS, respectively,
by courtesy of Dr. Y. Richard Yang and Dr. Zhiwei Xu.



Outline

- ❑ Admin and recap
- ❑ Java graphics (StdDraw)
- ❑ Parameterized graphics methods and loops

Recap: DrawX

1. Bound

• == , $2S - 4$ spaces, ==

2. for line = 1 to $S - 1$

line spaces

\

$2S - 2 - 2 * \text{line spaces}$

/

3. for line = 1 to $S - 1$

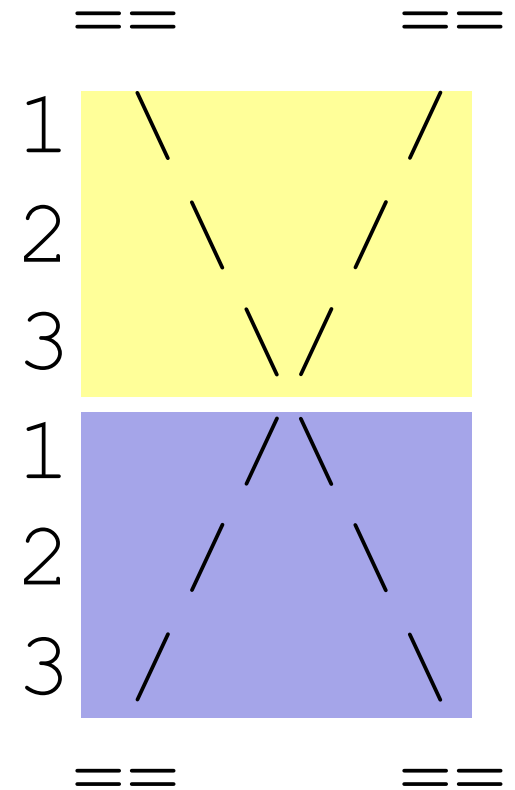
$S - \text{line spaces}$

/

$(\text{line} - 1) * 2$ spaces

\

4. Bound



Recap: Identify Subtasks (Methods)

1. Bound

• == $2S - 4$ spaces, ==

2. for line = 1 to $S - 1$

line spaces

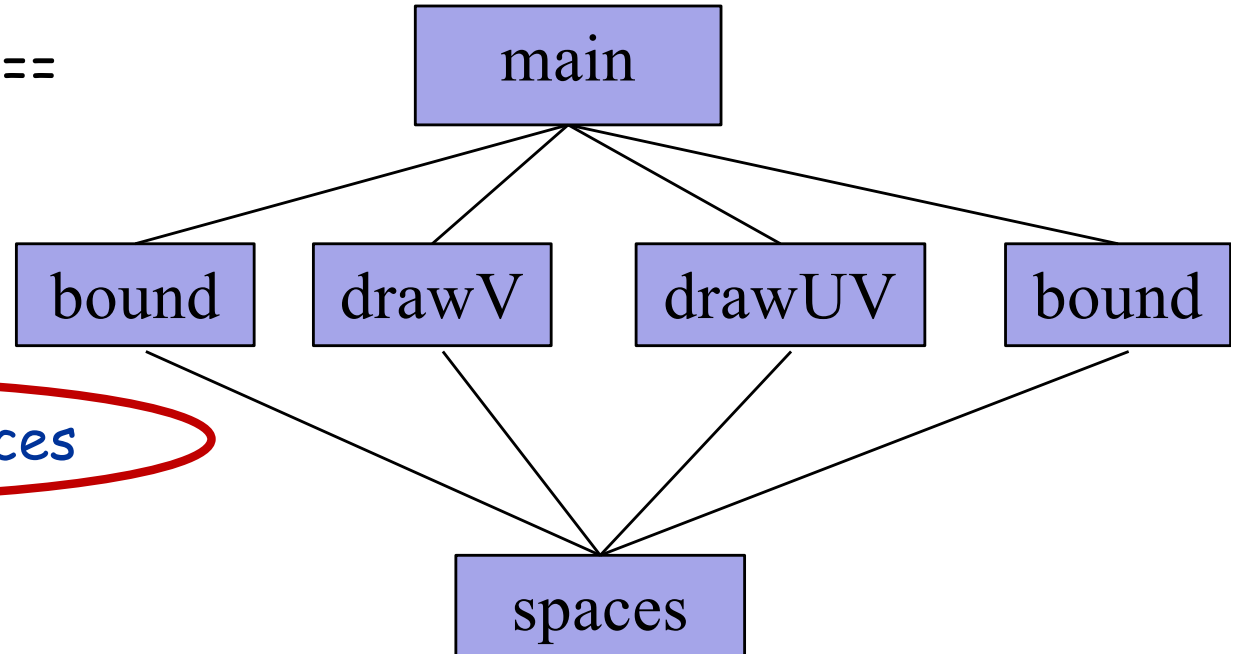
$2S - 2 - 2 * \text{line spaces}$

3. for line = 1 to $S - 1$

$S - \text{line spaces}$

$(\text{line} - 1) * 2$ spaces

4. Bound



Drawing spaces is a reusable function, but need parameter to be reusable.

Method w/ Parameters

- ❑ Why: Redundancy removal/abstraction through generalization
- ❑ General: a method can accept multiple parameters. (separate by ,)
 - When calling it, you must pass values for each parameter.

❑ Declaration:

```
public static void <name>(<type> <name>, ..., <type> <name>) {  
    <statement>(s);  
}
```

❑ Call:

```
<name>(<exp>, <exp>, ..., <exp>);
```

Multiple Parameters Example

```
public static void main(String[] args) {  
    printNumber(4, 9);  
    printNumber(17, 6);  
    printNumber(8, 0);  
    printNumber(0, 8);  
}  
  
public static void printNumber(int number, int count) {  
    for (int i = 1; i <= count; i++) {  
        System.out.print(number);  
    }  
    System.out.println();  
}
```

Output:

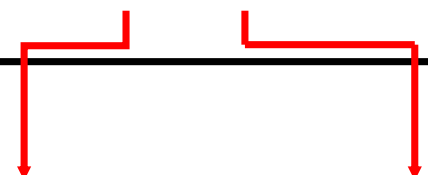
```
4444444444  
171717171717  
  
00000000
```

Multiple Parameter Invocation

- Corresponding actual *argument* in the invocation is copied into the corresponding *formal argument*

```
printNumber(1+3, 1+1);
```

```
public static void printNumber(int number, int count)
{
    // equiv: number = 4; count = 2;
    for (int i = 1; i <= count; i++) {
        System.out.print(number);
    }
    System.out.println();
}
```

A diagram consisting of two red arrows. The first arrow starts under the expression '1+3' in the invocation 'printNumber(1+3, 1+1);' and points down to the parameter 'number' in the method signature 'public static void printNumber(int number, int count)'. The second arrow starts under the expression '1+1' in the invocation and points down to the parameter 'count' in the method signature.

Exercise

- ❑ Define `displayChars` to display a char `ch` `count` times
- ❑ Implement `spaces` by calling `displayChars`

Foundational Programming Concepts

any program you might want to write

objects

methods and classes

graphics, sound, and image I/O

arrays

conditionals and loops

math

text I/O

primitive data types assignment statements

Java Graphics

- ❑ Java provides a large number of methods for graphics
- ❑ A graphical method may need to use a large number of parameters
 - E.g., draw a line: line color, stroke width, stroke pattern, init pos, end pos
- ❑ To avoid specifying a large of parameters in each method call, Java native graphics uses an object-oriented design: state parameters are contained (encapsulated) in entities called objects
 - We will cover objects in more details later

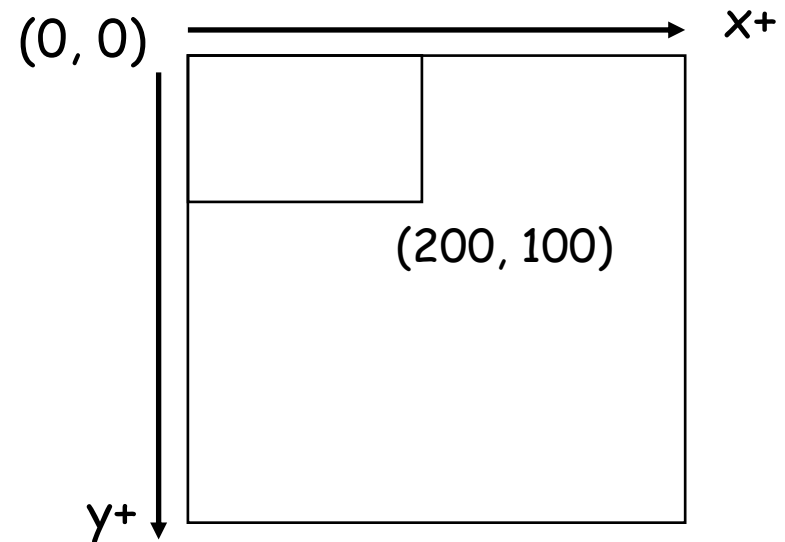
Java Graphics Wrapper Classes

- ❑ To simplify the usage of Java native graphics, wrapper graphics classes are provided
 - Back to Basics Textbook: defines class `DrawingPanel`, which is still an object-oriented design
 - Sedgewick & Wayne book: avoids objects, by defining a class called `StdDraw`, which contains many static (non-object-oriented) graphics methods:
 - <http://introcs.cs.princeton.edu/java/stdlib/javadoc/StdDraw.html>
 - To access a static method defined in a class, use `<class-name>.<method-name>(...)`, for example,
 - `StdDraw.line (0, 0, 10, 10);`
 - `DrawingPanel` and `StdDraw` are good examples that method designers may differ substantially in their designs, although for very similar functions

DrawingPanel Design (Back to Basics Textbook)

□ Coordinate system

- Each (x, y) position is *pixel* ("picture element") position
- Position $(0, 0)$ is at the window's **top-left** corner.
 - x increases rightward and the y increases downward.



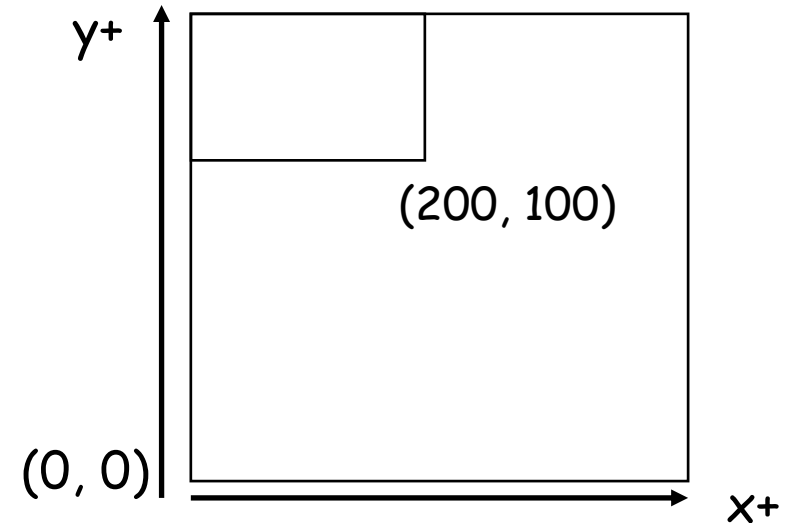
□ Example method rectangle:

- DrawingPanel: `rect (int x0, int y0, int width, int height)`,
 - e.g., `rect(0, 0, 200, 100)`

StdDraw (The One we will use)

□ Coordinate system

- You still set canvas size using numbers of pixels:
`setCanvasSize(int w, int h)`
- But (x, y) position is the coordinate in a **normalized** coordinate system
 - [0, 1] as default
 - `setXScale(double x0, double x1)` to set x range
 - `setYScale(double y0, double y1)`
- Position (0, 0) is at the window's **lower-left** corner.
 - x increases rightward and the y increases upward.



```
StdDraw.setCanvasSize(500, 500);  
StdDraw.setXScale(0, 500);  
StdDraw.setYScale(0, 500);  
StdDraw.rectangle(100, 450,  
                  100, 50);
```

□ Example method rectangle:

- `rectangle(double cx, double cy, double halfWidth, double halfHeight);`

Example StdDraw Methods

<code>void line(double x0, double y0, double x1, double y1)</code>	
<code>void point(double x, double y)</code>	
<code>void text(double x, double y, String s)</code>	
<code>void circle(double x, double y, double r)</code>	
<code>void filledCircle(double x, double y, double r)</code>	
<code>void square(double x, double y, double r)</code>	
<code>void filledSquare(double x, double y, double r)</code>	
<code>void polygon(double[] x, double[] y)</code>	
<code>void filledPolygon(double[] x, double[] y)</code>	
<code>void setXscale(double x0, double x1)</code>	<i>reset x range to (x_0, x_1)</i>
<code>void setYscale(double y0, double y1)</code>	<i>reset y range to (y_0, y_1)</i>
<code>void setPenRadius(double r)</code>	<i>set pen radius to r</i>
<code>void setPenColor(Color c)</code>	<i>set pen color to c</i>
<code>void setFont(Font f)</code>	<i>set text font to f</i>
<code>void setCanvasSize(int w, int h)</code>	<i>set canvas to w-by-h window</i>
<code>void clear(Color c)</code>	<i>clear the canvas; color it c</i>
<code>void show(int dt)</code>	<i>show all; pause dt milliseconds</i>
<code>void save(String filename)</code>	<i>save to a .jpg or w.png file</i>

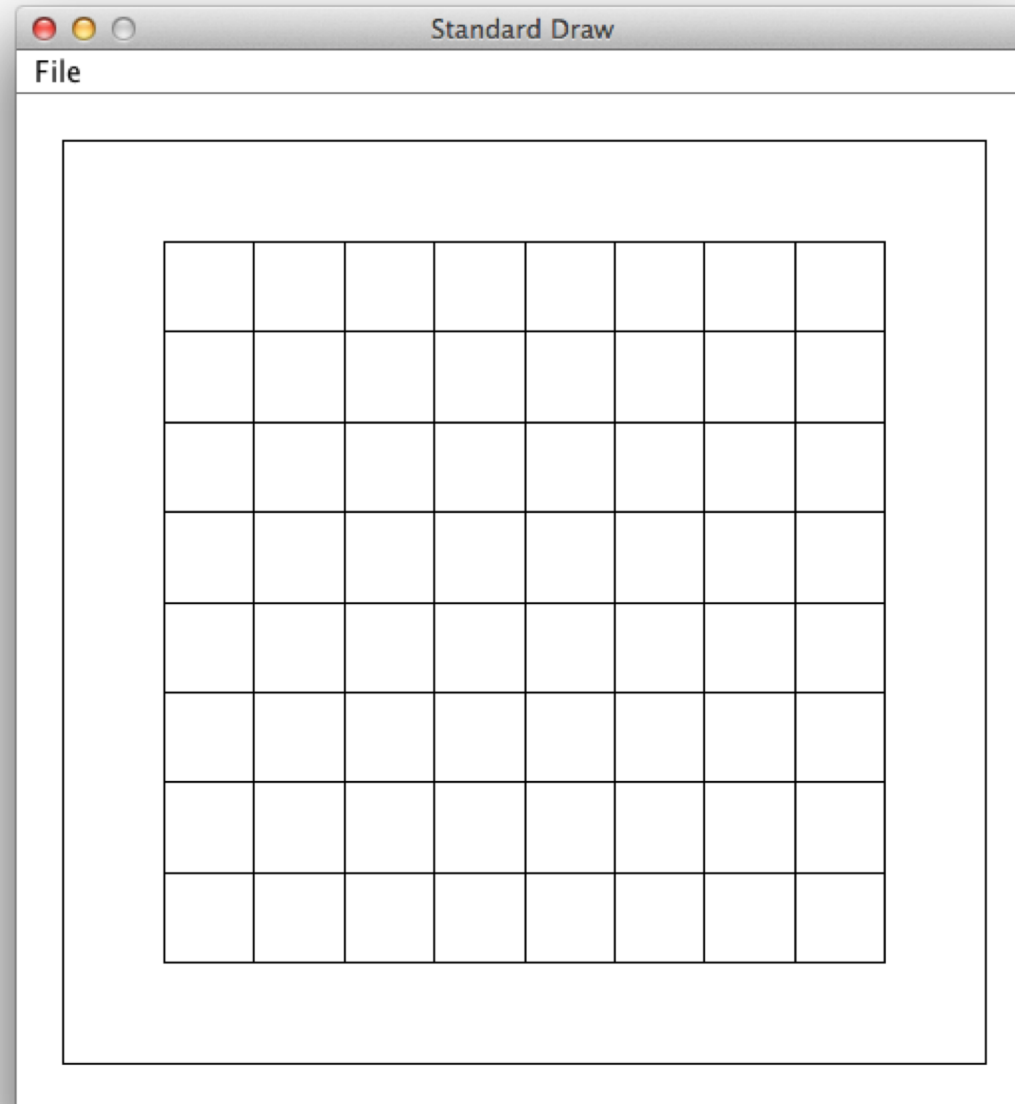
Example: StdDraw X

- ❑ Implement SimpleStdDrawX.java to draw an X using StdDraw
 - Try two approaches:
 1. Use the normalized coordinate system (0,0) -> (1,1)
 2. Set the Xscale and Yscale

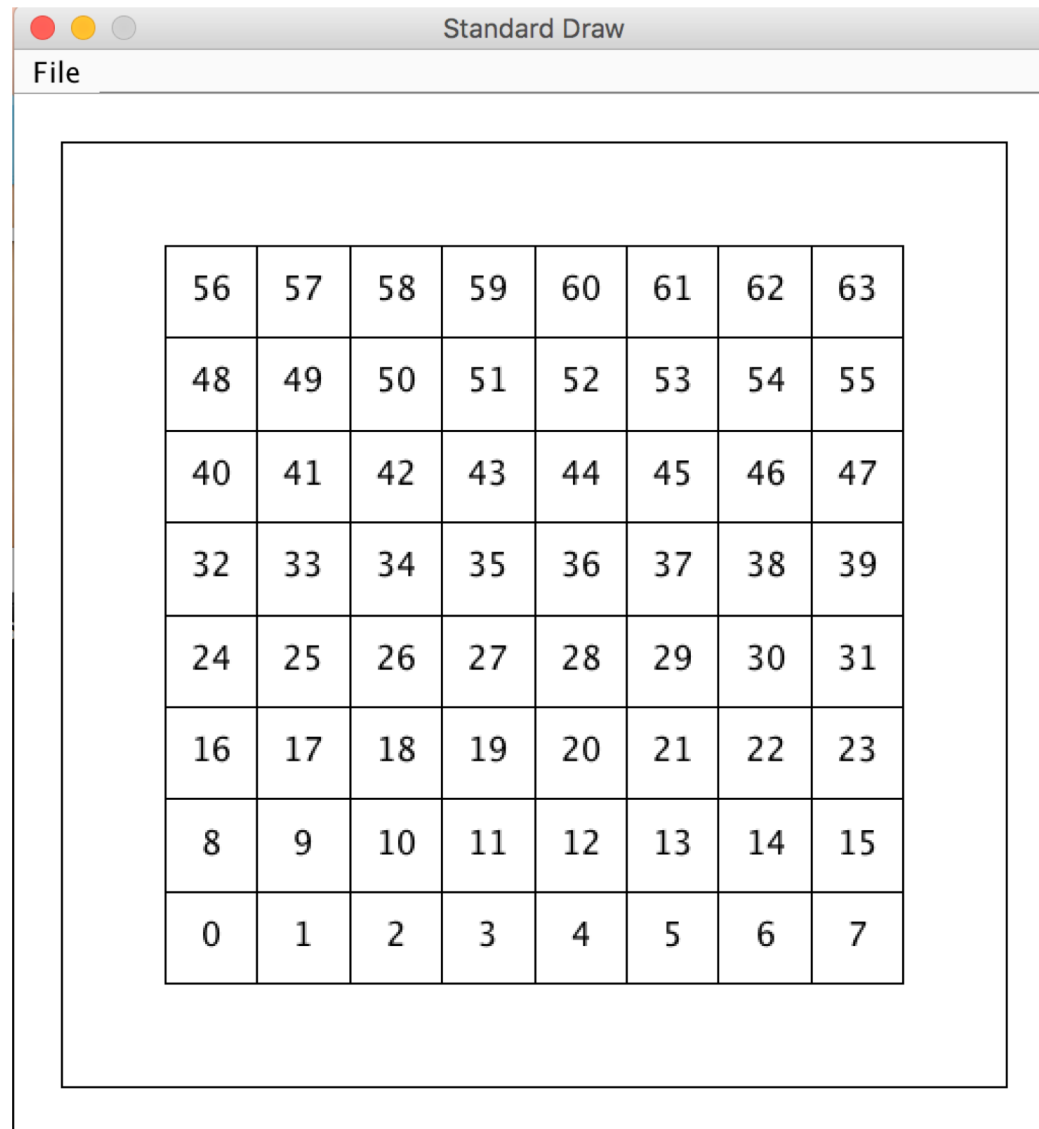
Example: StdDraw Loops

- ❑ You may use loop (nested loop) to produce more "complex" patterns.
- ❑ See SimpleStdDrawLoop.java and predict its display

Example: Draw using Loops



Example: Modify SimpleStdDrawLoop to Label a Number for each Cell



Standard Draw

File

56	57	58	59	60	61	62	63
48	49	50	51	52	53	54	55
40	41	42	43	44	45	46	47
32	33	34	35	36	37	38	39
24	25	26	27	28	29	30	31
16	17	18	19	20	21	22	23
8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7

Outline

- ❑ Admin and recap
- ❑ Java graphics
 - Coordinate system and basic shape
 - Drawing w/ color

Example: Draw with Color

- ❑ What if we want to draw the X in red?
 - StdDraw has two methods to set pen color
 - StdDraw.setPenColor(R, G, B);
 - StdDraw.setPenColor(Color);
 - predefined **class constants** defined in the `Color` class

Exercise: SimpleStdDrawXColor

- ❑ Modify SimpleStdDrawX to use `setPenColor(R, G, B)` to draw the lines red

Class Constants

- ❑ **class constant:** A static class variable with a fixed value
 - value can be set only at declaration; cannot be reassigned

- ❑ **Syntax:**

`public static final type name = value; // in class scope`

- name is usually in `ALL_UPPER_CASE`

- **Examples:**

```
public static final int DAYS_IN_WEEK = 7;
public static final double INTEREST_RATE = 3.5;
public static final int SSN = 658234569;
```

Color



- Java predefines many **class constants** in the `Color` class:

`Color.CONSTANT_NAME`

where **CONSTANT_NAME** is one of:

BLACK,	BLUE,	CYAN,	DARK_GRAY,
	GRAY,		
GREEN,	LIGHT_GRAY,	MAGENTA,	ORANGE,
PINK,	RED,	WHITE,	YELLOW

<http://download.oracle.com/javase/8/docs/api/java/awt/Color.html>

Complexity of Using the `Color` Class: Class Library

- ❑ The `Color` class is part of **Java standard class library**
- A class library is a collection of classes that one can use when developing programs
 - libraries are not part of the Java language per se, but using them is essential to achieve productive programming
 - A big advantage of Java is that it provides a quite large standard class library

Library and Packages

- ❑ The classes in a library are organized into **packages**
 - think of packages as folders, which help you to get organized
- ❑ Some of the packages in the standard class library are:

<u>Package</u>	<u>Purpose</u>
<code>java.lang</code>	General support, e.g., Math, String, System
<code>java.applet</code>	Creating applets for the web
<code>java.awt</code>	Graphics and graphical user interfaces
<code>javax.swing</code>	Additional graphics capabilities and components
<code>java.net</code>	Network communication
<code>java.util</code>	Utilities
<code>java.text</code>	Text processing

- ❑ `Color` belongs to a package named `java.awt`

<http://download.oracle.com/javase/7/docs/api/>

The import Declaration

- ❑ When you want to use a class from a package, you could use its *fully qualified class name, e.g.,*

```
java.awt.Color color;
```

Or you can *import* the class, then just use the class name

```
// put this at the very top of your program  
import java.awt.Color;
```

- ❑ To import all classes in a particular package, you can use the * wildcard character

```
// put this at the very top of your program  
import java.awt.*;
```

Example: Using Colors

- ❑ Pass a `Color` to `StdDraw's` `setPenColor` method
 - Subsequent shapes will be drawn in the new color.

```
import java.awt.Color;
StdDraw.setPenColor(Color.BLUE);
StdDraw.line(20, 0, 10, 30);
```

```
StdDraw.setPenColor(Color.RED);
StdDraw.line(20, 0, 10, 30);
```

Exercise: SimpleStdDrawX

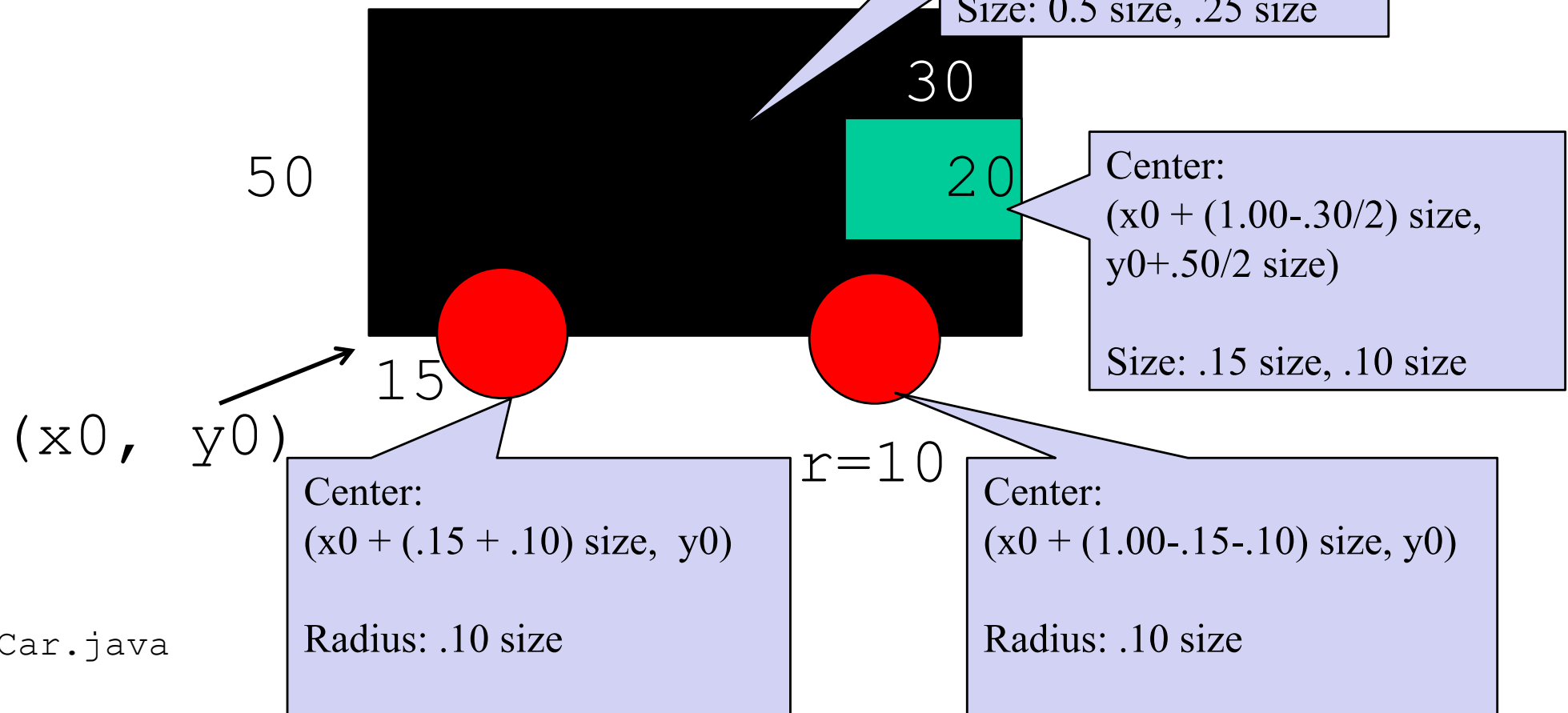
- ❑ Modify SimpleStdDrawX to draw the lines red using Color.RED

Outline

- ❑ Admin and recap
- ❑ Java graphics
 - Coordinate system and basic shape
 - Drawing w/ color
 - Parameterized drawing examples

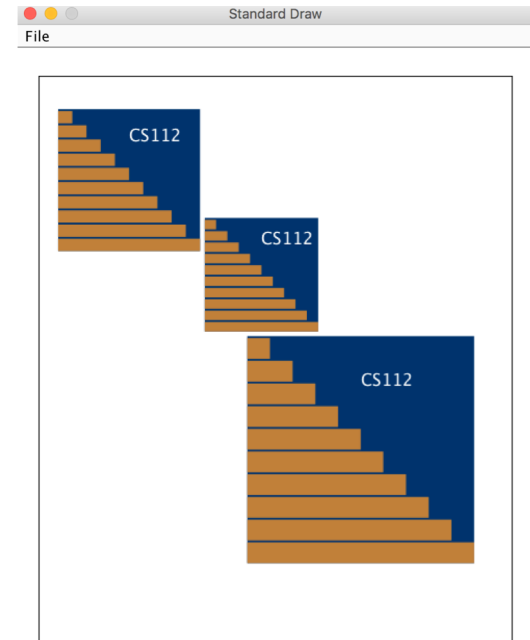
Exercise: Parameterized Drawing

- Write method
`drawCar(x0, y0, size):`
`size=100`



Exercise: Book Cover (Color and Loop)

- ❑ White 500x600 drawing panel
- ❑ Three components at
 - (20, 415), (165, 330), (220, 85) with sizes 150, 120, and 240
 - Each component
 - XMU blue background
 - white “CT-F25” text left @ 1/2 horizontal, 4/5 vertical
 - 10 brown (red=192, green=128, blue=64) “bricks”
 - 2 pixel between two adjacent bricks



Outline

- ❑ Admin and recap
- ❑ Animations

Question: Car Launch



- ❑ Two cars launching from different heights, at different speeds, compute their positions in the initial 10 sec.



Car 1: One World Trade Center

- * initial height: 541 m
- * speed (210 km/h), with
horizontal speed: 50 m/sec
vertical speed: 30 m/sec



Car 2: Empire State

- * initial height: 381 m
- * speed (114 km/h), with
horizontal speed: 30 m/sec
vertical speed: 10 m/sec

Background: Physics

- ❑ In physics, for each dimension (x or y), given initial velocity v_0 in m/s, acceleration a in m/s^2 , and elapsed time t in s
 - the speed of the moving body at time t :
 - $V = v_0 + a t$
 - The *displacement* of the moving body at time t :
 - $\text{Displacement} = v_0 t + \frac{1}{2} a t^2$
 - The position of the moving body at time t :
 - $\text{Pos} = \text{initial position} + \text{displacement}$



Position

- $p_0 + v_0 t + \frac{1}{2} a t^2$

- Horizontal (x):

- $v_{x0} * t$

- Vertical (y):

- $g \cong 9.81 \text{ m/s}^2$, downward

- $h_0 + v_{y0} t - \frac{1}{2} g t^2$

CarLaunch: Initial Version

```
// You must have StdAudio.java and race-car.wav in the  
// same directory and first compile StdAudio.java.
```

```
StdAudio.loop("race-car.wav");
```

You can use StdAudio
to add sound

```
// set up the initial state of the two cars
```

```
int h1 = 541, v1x = 50, v1y = 30;
```

```
int h2 = 381, v2x = 30, v2y = 10;
```

```
// Simulate time from 0 to 10 sec.
```

```
for (double t = 0; t < 10; t += 0.1) {
```

```
    // Compute car 1's position
```

```
    double x1 = v1x * t;
```

```
    double y1 = h1 + v1y * t - 0.5 * 9.81 * t * t;
```

```
    // Compute car 2's position
```

```
    double x2 = v2x * t;
```

```
    double y2 = h2 + v2y * t - 0.5 * 9.81 * t * t;
```

```
    // Use the method defined in Car.java
```

```
    StdDraw.picture(x1, y1, "angry-bird-b.png");
```

```
    Car.drawCar(x2, y2, CAR2_SIZE);
```

```
} // end of for
```

One can mix param.
drawing and fixed
image file

CarLaunch: No Trace

```
// You must have StdAudio.java and race-car.wav in the
// same directory and first compile StdAudio.java.
StdAudio.loop("race-car.wav");

// set up the initial state of the two cars
int h1 = 541, v1x = 50, v1y = 30;
int h2 = 381, v2x = 30, v2y = 10;

// Simulate time from 0 to 10 sec.
for (double t = 0; t < 10; t += 0.1) {
    // Compute car 1's position
    double x1 = v1x * t;
    double y1 = h1 + v1y * t - 0.5 * 9.81 * t * t;

    // Compute car 2's position
    double x2 = v2x * t;
    double y2 = h2 + v2y * t - 0.5 * 9.81 * t * t;

    // Use the method defined in Car.java
    Car.drawCar(x1, y2, CAR1_SIZE);
    Car.drawCar(x2, y2, CAR2_SIZE);

    StdDraw.clear();
} // end of for
```

Removes all content
from display

CarLaunch: using StdDraw.show(t) for Timing

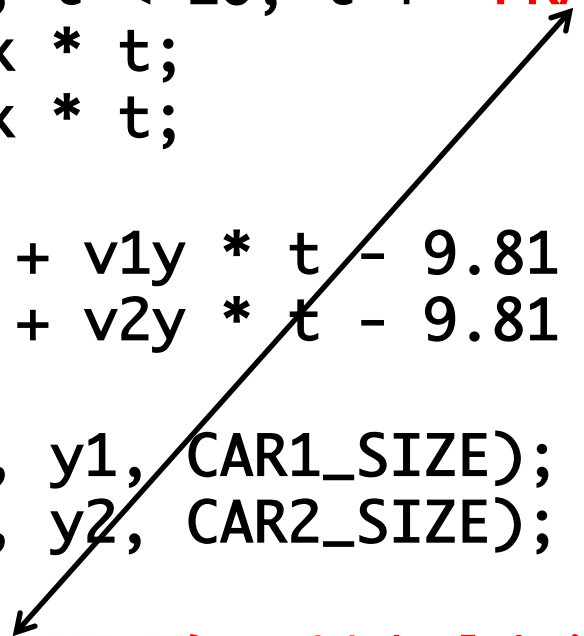
```
// FRAME_T = 60;
int h1 = 541, v1x = 50, v1y = 30;
int h2 = 381, v2x = 30, v2y = 10;

for (double t = 0; t < 10; t += FRAME_T/1000.0 ) {
    double x1 = v1x * t;
    double x2 = v2x * t;

    double y1 = h1 + v1y * t - 9.81 * t * t / 2;
    double y2 = h2 + v2y * t - 9.81 * t * t / 2;

    Car.drawCar(x1, y1, CAR1_SIZE);
    Car.drawCar(x2, y2, CAR2_SIZE);

    StdDraw.show(FRAME_T); // hold image for 60 ms
    StdDraw.clear(); // now clear up
}
```

A black arrow originates from the expression `FRAME_T/1000.0` in the loop increment `t += FRAME_T/1000.0` and points to the argument `FRAME_T` in the `StdDraw.show(FRAME_T);` statement, illustrating that the loop increment represents the time interval in seconds corresponding to the frame time in milliseconds.

StdDraw.show(t) and Animation

□ StdDraw.show is overloaded

- StdDraw.show(int t)
 - Display on screen, pause for t milliseconds, and **turn on animation mode**: subsequent calls to drawing methods such as line(), circle(), and square() will not be displayed on screen until the next call to StdDraw.show.
- StdDraw.show()
 - Display on-screen and **turn off animation mode**: subsequent calls to drawing methods such as line(), circle(), and square() will be displayed on screen when called.

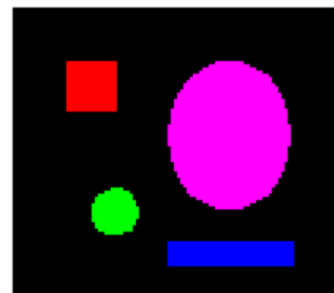
On-Screen

- ❑ The display of your computer is just a mirror of the content of the Video Memory

Video Memory



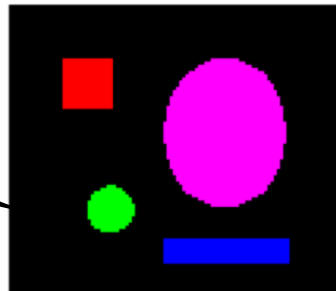
Video Memory



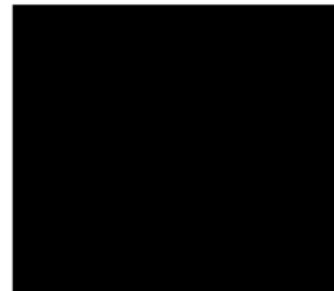
On-Screen (Video Memory) and (Double) Buffer

In animation mode, drawing is in the (double) buffer

Double Buffer in System Memory

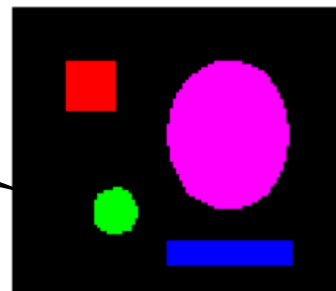


Video Memory

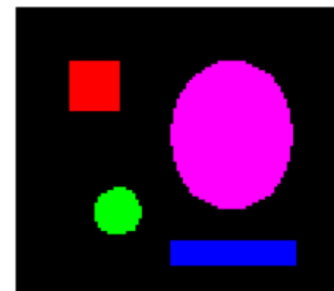


Next show(t) statement copies to screen

Double Buffer in System Memory



Video Memory



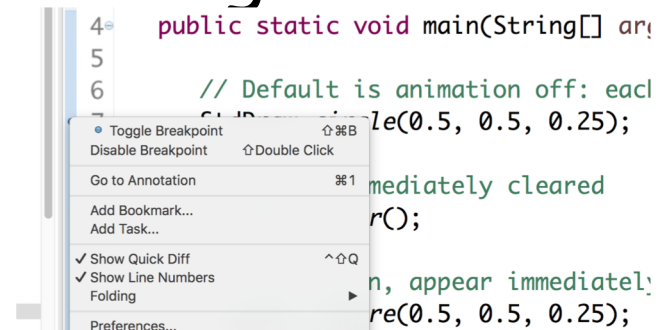
When finished drawing, the double buffer is copied to video memory.



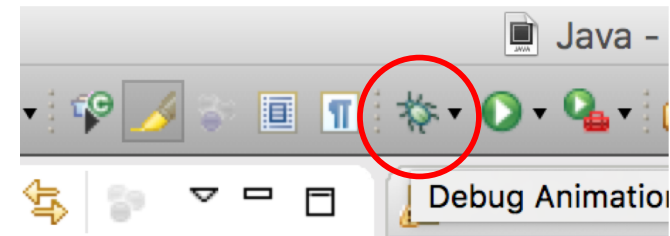
Example: AnimationShow.java

❑ Try running the example in Debug mode

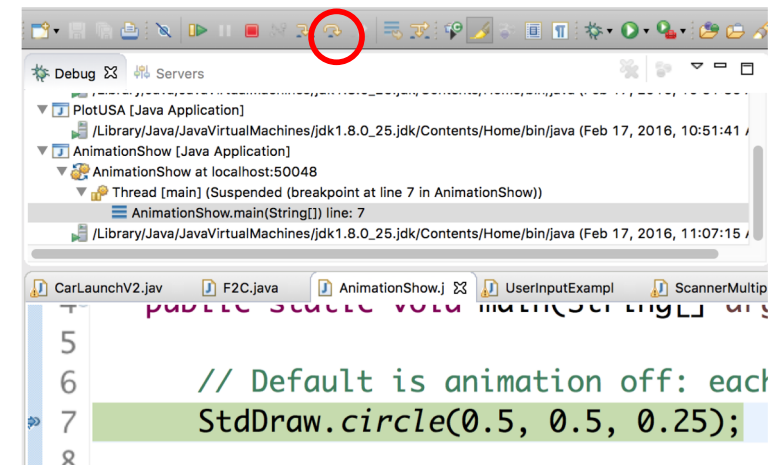
- Right click first line and toggle breakpoint



- Execute in Debug mode



- Execute (step over) each statement



Exercise: Add a Countdown Scene

- ❑ Count down from 10 to 0 and then start the race

Exercise: Add a Countdown Scene

- ❑ Count down from 10 to 0 and then start the race

```
public static void sceneStart(int h1, int h2) {  
    for (int t = 10; t >= 0; t--) {  
        Car.drawCar(0, h1, CAR1_SIZE);  
        Car.drawCar(0, h2, CAR2_SIZE);  
  
        StdDraw.text( WIDTH/2, HEIGHT/2, ""+t );  
  
        StdDraw.show( 1000 );  
  
        StdDraw.clear();  
    }  
}
```