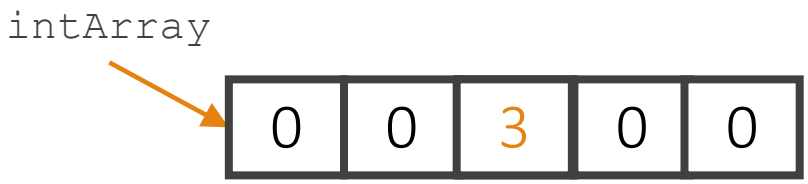

LELAND STANFORD JUNIOR UNIVERSITY
DIE LUFT DER FREIHEIT WEHT
1891

ArrayLists

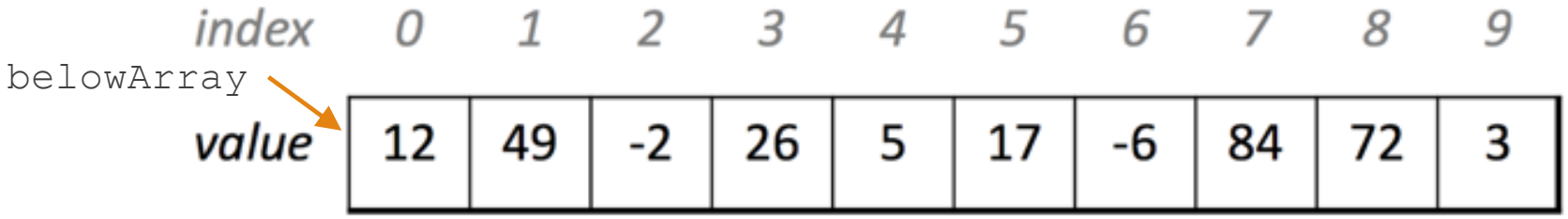
Previously…

- An **array** is a variable type that represents a list of items.
- You access individual items in an array by *index*.
- Stores a single type of item (**`int`**, **`double`**, **`GRect`**, etc.)

```
int[] intArray = new int[5];
intArray[2] = 3;
```



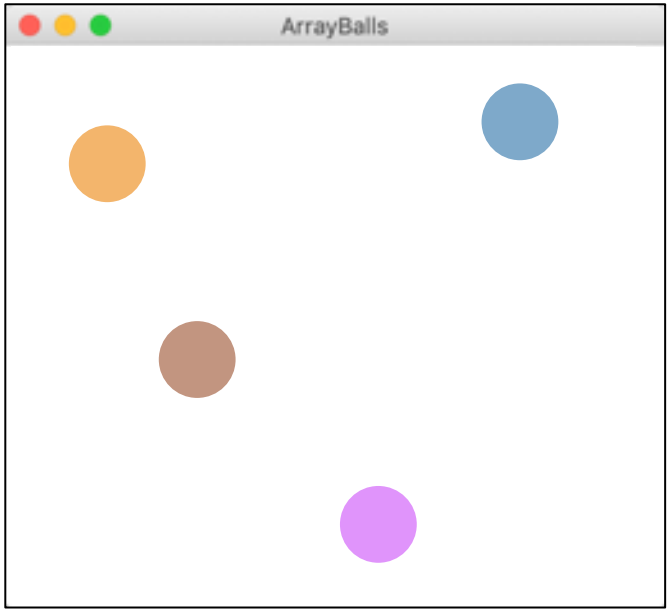

```
int[] belowArray = {12, 49, -2, 26, 5, 17, -6, 84, 72, 3};
```



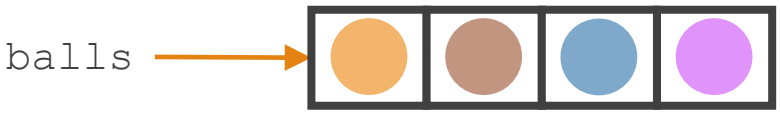
A quick warmup

How do we program the following:

- 4-element GOval (50x50) array
- Random colors
- Put in random place on canvas



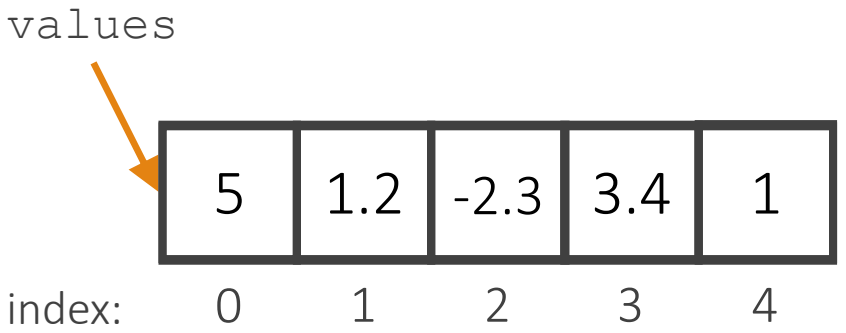

```
GOval[] balls = new GOval[N_BALLS];
for(int i = 0; i < balls.length; i++) {
    balls[i] = new GOval(BALL_SIZE, BALL_SIZE);
    balls[i].setFilled(true);
    balls[i].setColor(rgen.nextColor());
    add(balls[i],
        rgen.nextDouble(0, getWidth() - BALL_SIZE),
        rgen.nextDouble(0, getHeight() - BALL_SIZE));
}
```



A Different User Experience

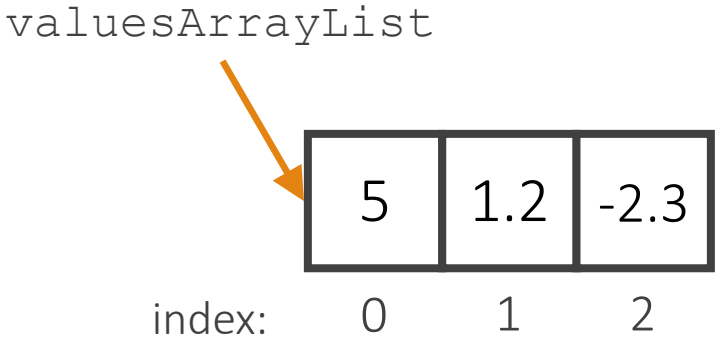
Your array program

```
How many values would you like to input? 5
Specify input for index 0 :1.2
Specify input for index 1 :-2.3
Specify input for index 2 : 3.4
Specify input for index 3 :1
Specify input for index 4 :0
```

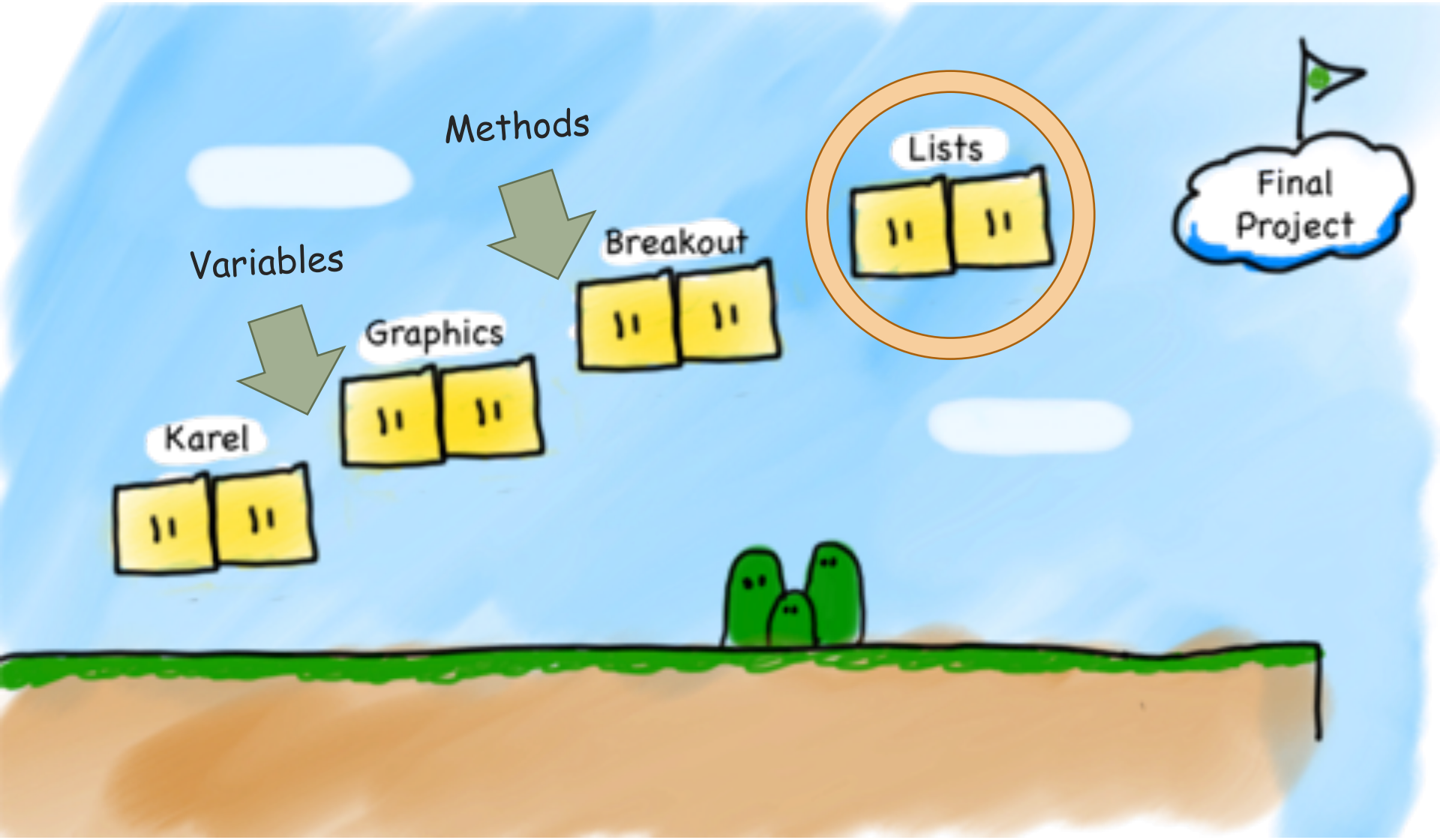


An ArrayList program

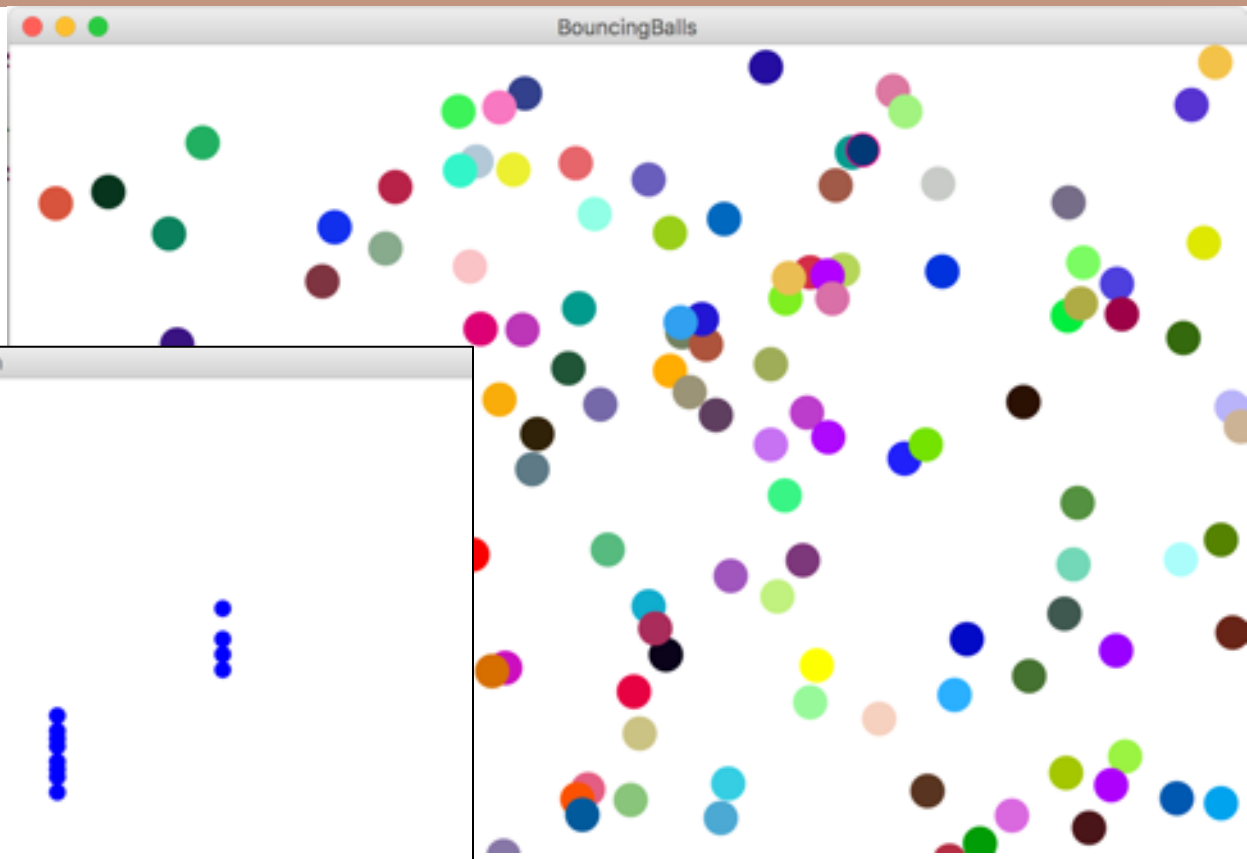
```
This program computes statistics.
Enter *nonzero* input (or 0 to end): -5
Enter *nonzero* input (or 0 to end): 3.1415926535
Enter *nonzero* input (or 0 to end): 3
Enter *nonzero* input (or 0 to end): 0
Your array: -5.0 3.1415926535 3.0 0.0
```

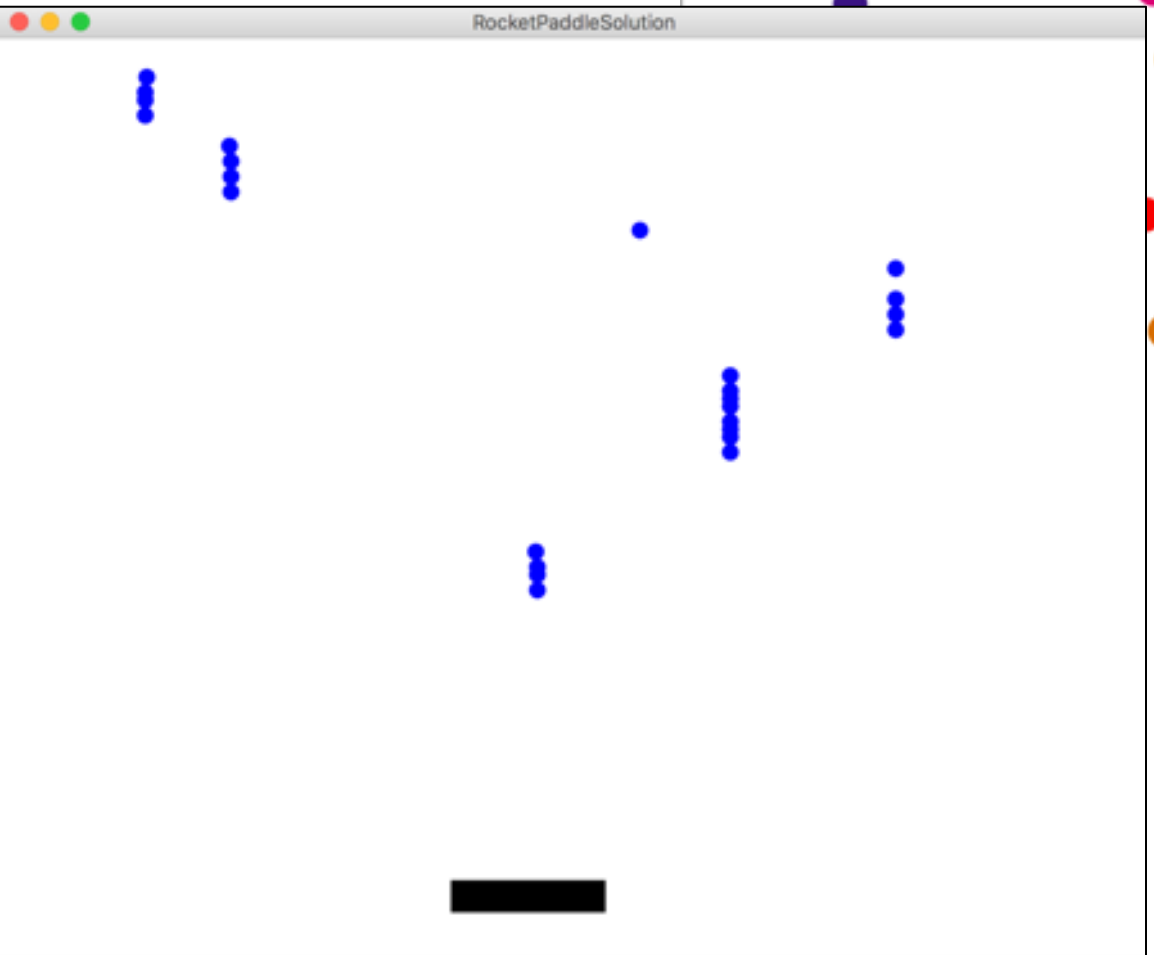


Our Penultimate Step



After This Lecture!





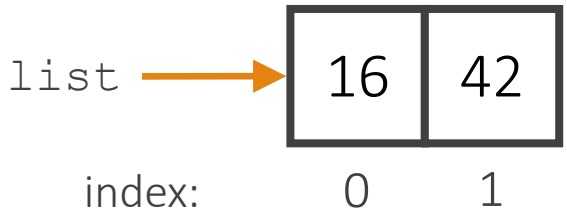
Meet ArrayLists

- A variable type that represents a list of items.
- You access individual items by index.
- Store a single type of object (String, GRect, etc.)
- *Resizable* – can add and remove elements
- Has helpful methods for searching for items

Memnun oldum!

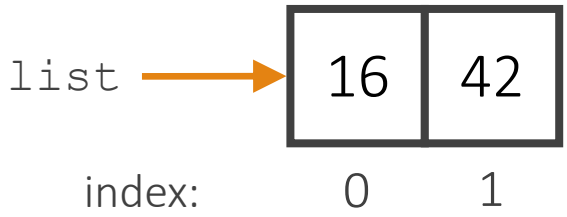
ArrayList

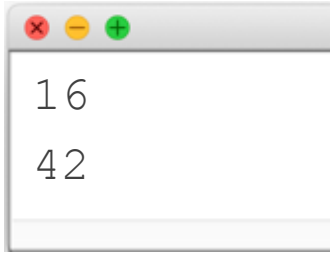
```
// Create an (initially empty) list
ArrayList <Integer> list = new ArrayList<Integer>();

// Add an element to the back
list.add(16); // now size 1
list.add(42); // now size 2
```



ArrayList

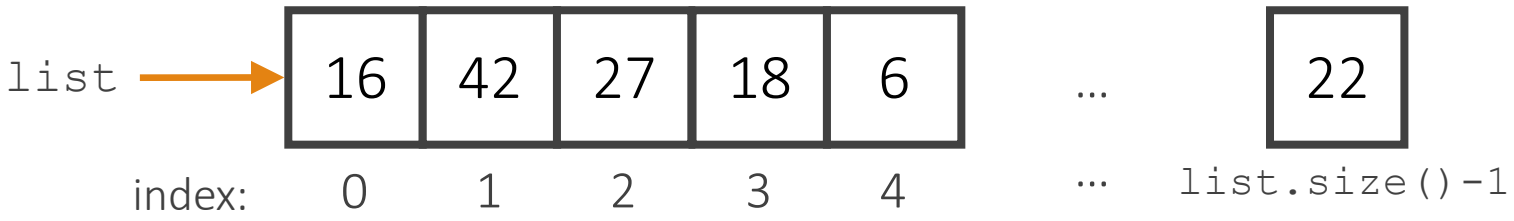
```
// Create an (initially empty) list
ArrayList <Integer> list = new ArrayList<Integer>();

// Add an element to the back
list.add(16); // now size 1
list.add(42); // now size 2

// Access elements by index (starting at 0!)
println(list.get(0)); // prints 16
println(list.get(1)); // prints 42
```





Looping over all elements

```
// Access elements by index (starting at 0!)
for (int i = 0; i < list.size(); i++) {
    println(list.get(i));
}
```

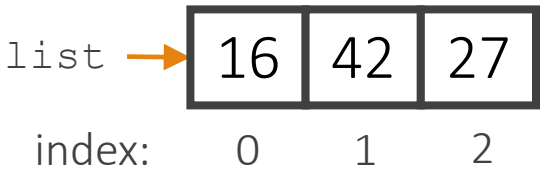


ArrayList Methods

`list.add(value);`	appends value at end of list
`list.add(index, value);`	inserts given value just before the given index, shifting subsequent values to the right
`list.clear();`	removes all elements of the list
`list.get(index)`	returns the value at given index
`list.indexOf(value)`	returns first index where given value is found in list (-1 if not found)
`list.isEmpty()`	returns `true` if the list contains no elements
`list.remove(index);`	removes/returns value at given index, shifting subsequent values to the left
`list.remove(value);`	removes the first occurrence of the value, if any
`list.set(index, value);`	replaces value at given index with given value
`list.size()`	returns the number of elements in the list
`list.toString()`	returns a string representation of the list such as `"[3, 42, -7, 15]"`

Insert/Remove

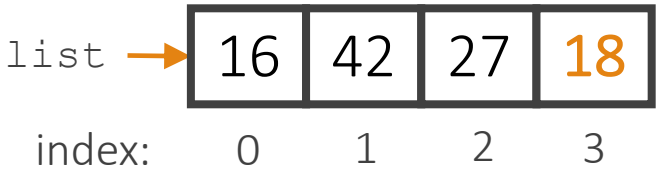
Original ArrayList:



If you insert/remove in the front or middle of a list, elements *shift* to fit.

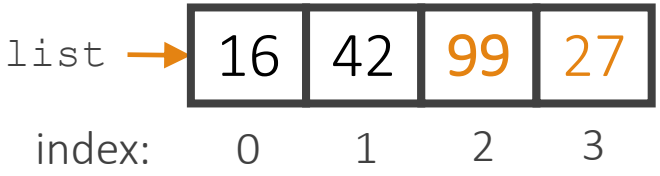
- Add element to end of list

```
list.add(18);
```



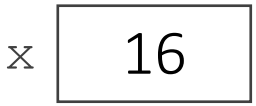
- Add element to middle of list

```
list.add(2,99);
```



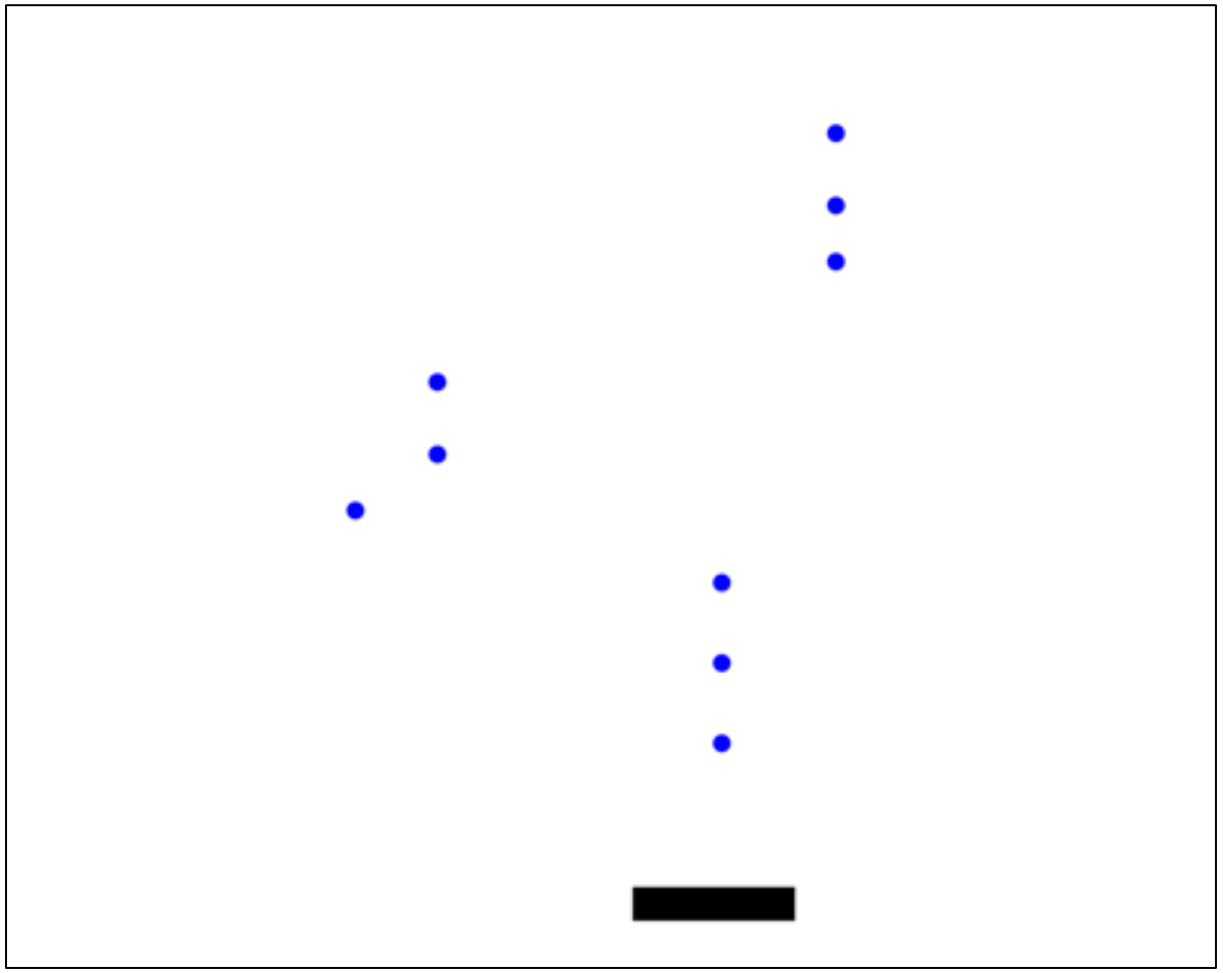
- Remove element from front of list

```
int x = list.remove(0);
```

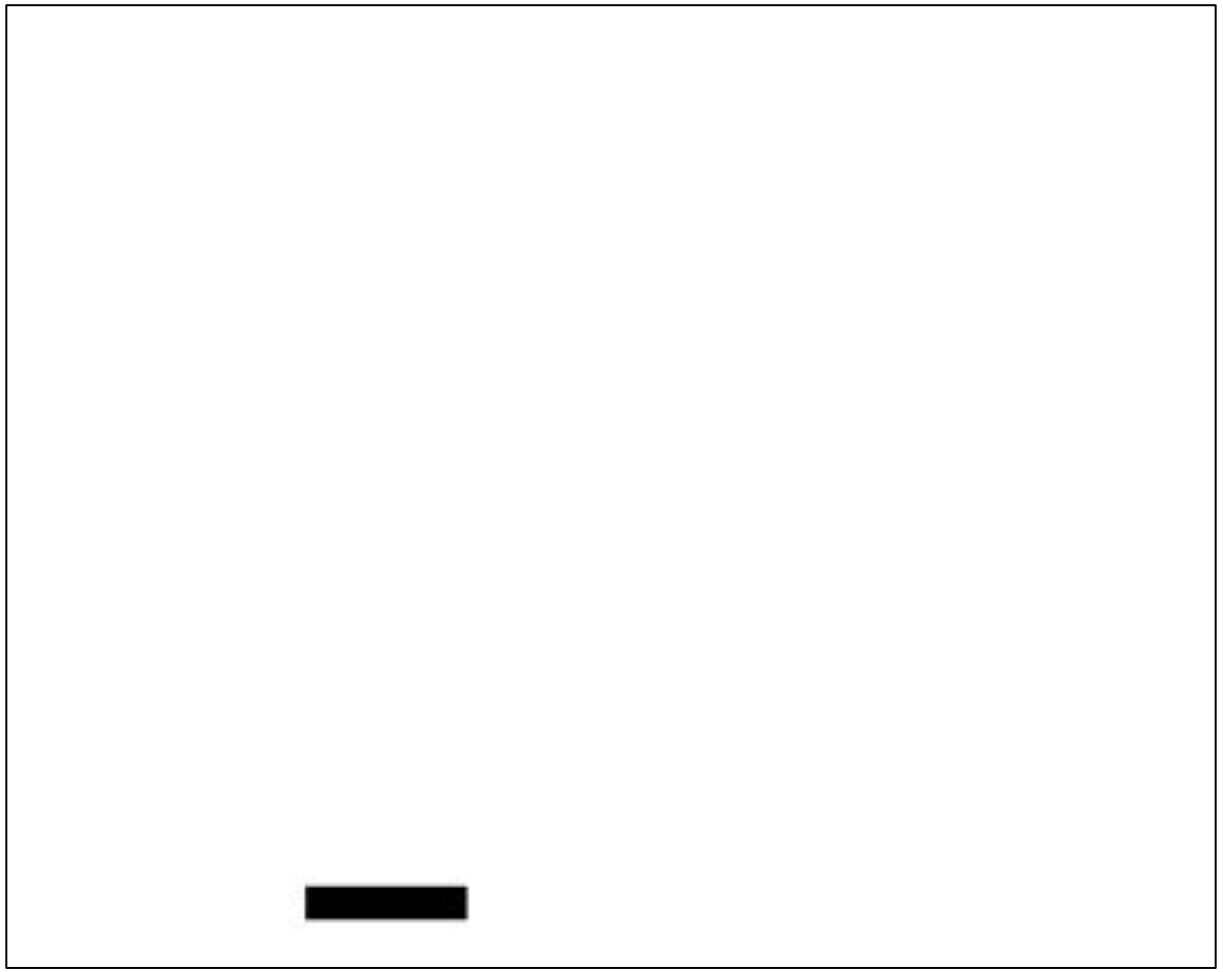


Questions?

Rocket Paddle



Rocket Paddle



`rocketList`: the ***visible*** rockets on the canvas

Rocket Paddle

```
import java.util.ArrayList;

public class RocketPaddle extends GraphicsProgram {

    private ArrayList<GOval> rocketList;
    private GRect paddle;

    public void run() {
        rocketList = new ArrayList<GOval>();
        createPaddle();
        addMouseListeners();

        while(true) {
            animateRockets();
            pause(100);
        }
    }
    ...
}
```

Java ArrayList library

Setup

Animate

rocketList: the *visible* rockets on the canvas

Interact

```
public void mousePressed(MouseEvent e) {
    double x = e.getX();
    double y = PADDLE_Y;
    GOval rocket = new GOval(x, y, BALL_SIZE, BALL_SIZE);
    ...
    add(rocket);                // add the rocket to the screen
    rocketList.add(rocket); // add the rocket to the list
}
```

Animate

```
private void animateRockets() {
    // loop over list backwards so that we can
    // safely remove from the list.
    for(int i = rocketList.size() - 1; i >= 0; i--) {
        GOval rocket = ?????? // get the rocket
        ?????? // move the rocket
        // remove the rocket
        ??????
    }
}
```

`rocketList`: the ***visible*** rockets on the canvas

ArrayLists and Primitives

```
// Doesn't compile ☹
ArrayList <int> list = new ArrayList<int>();
```

2x

Syntax error, insert "Dimensions" to complete ReferenceType

Unlike arrays, ArrayLists can only store *objects*.

GRect

GOval

String

ArrayLists and Primitives

```
// Doesn't compile ☹
ArrayList <int> list = new ArrayList<int>();
```

2x

Syntax error, insert "Dimensions" to complete ReferenceType

Unlike arrays, ArrayLists can only store *objects*.

Primitive	"Wrapper" Class
`int`	`Integer`
`double`	`Double`
`boolean`	`Boolean`
`char`	`Character`

Objects: `GRect`, `GOval`, `String`, etc.

ArrayLists and Primitives

```
// Doesn't compile ☹
ArrayList <int> list = new ArrayList<int>();
```

```
// Use wrapper classes when making an ArrayList
ArrayList <Integer> list = new ArrayList<Integer>();

// Java converts Integer <-> int automatically!
int num = 123;
list.add(num);

int first = list.get(0); // 123
```

ArrayLists vs. Arrays

ArrayLists

(+) Can add/remove elements
(–) Needs wrapper class for primitives

Good for:

Lists updated through user interaction

Arrays

(+/–) Fixed size
(+) Simpler syntax
(+) Multi-dimensional arrays! (images)

Good for:

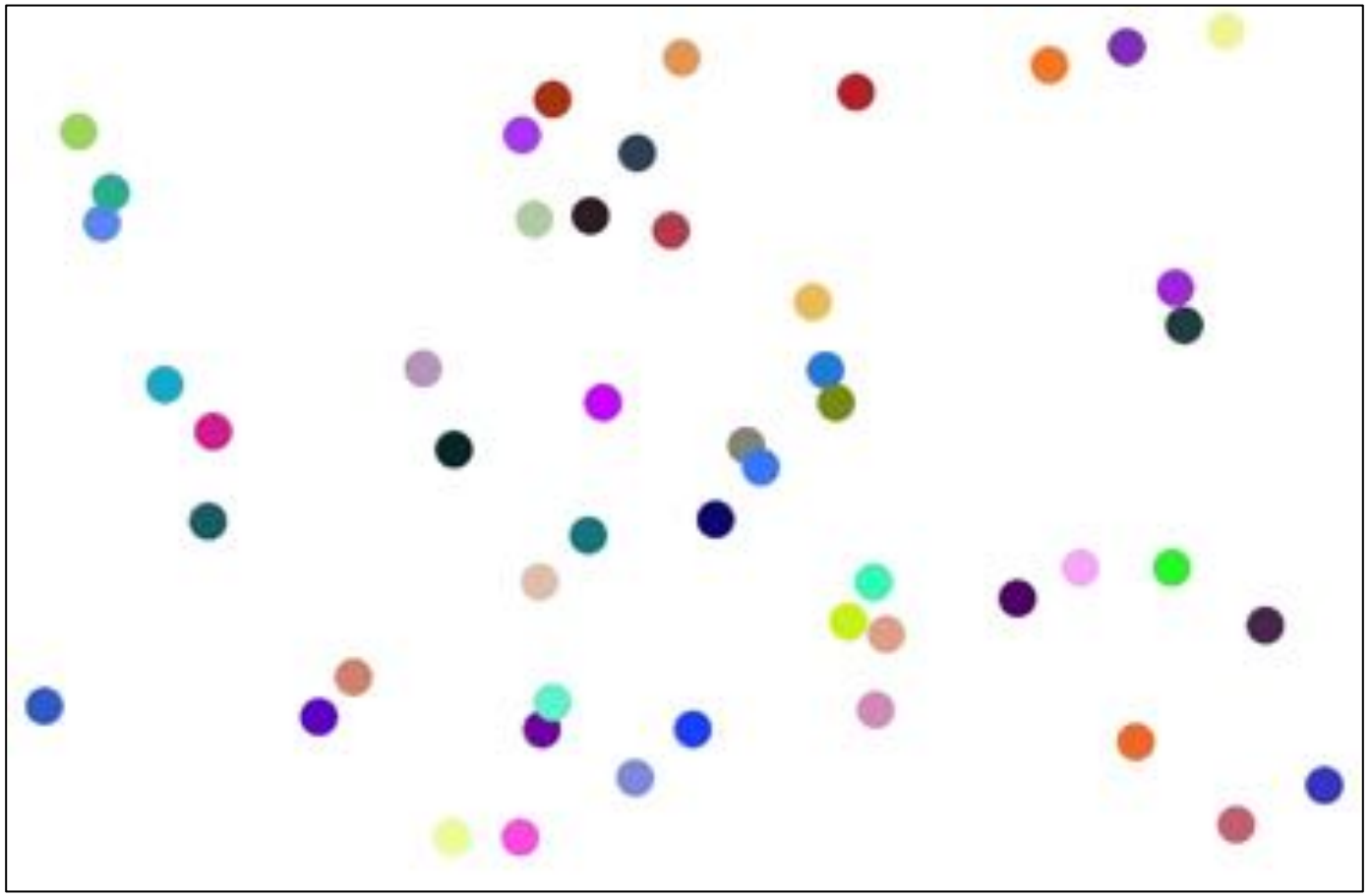
Constant list for lookup
Updating a grid

Why do both of these exist in the Java language?

- Arrays are Java's fundamental data storage
- `ArrayList` is a library built on top of an array

Questions?

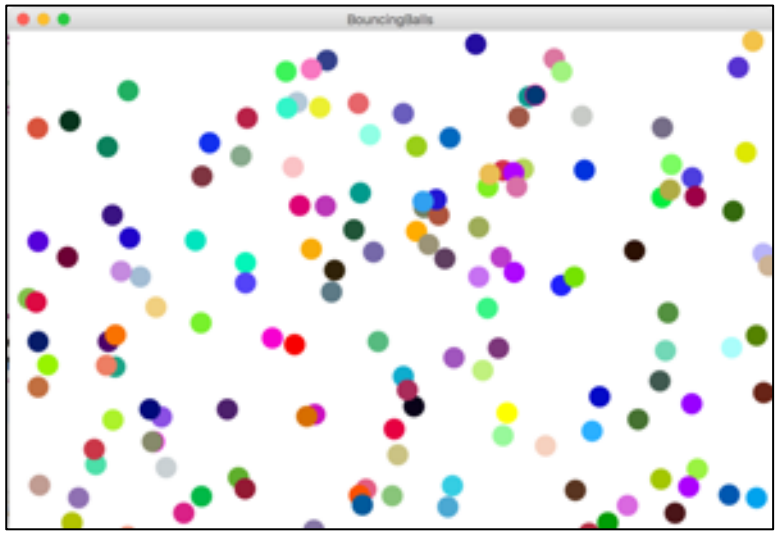
Bouncing Balls



Implementation ideas for Final Project!

(example) BouncingBalls.java

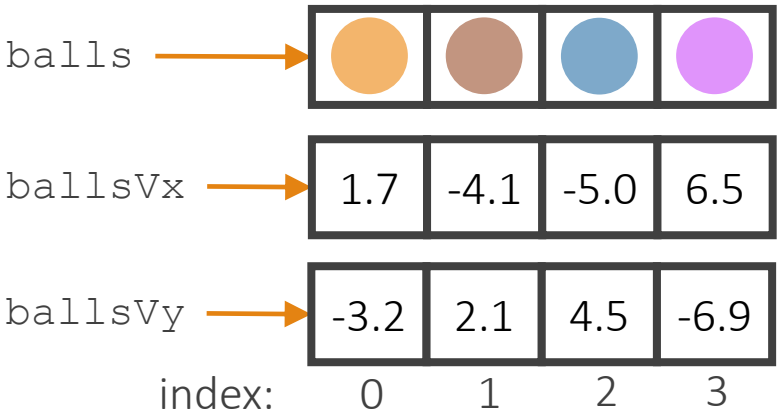
Bouncing Balls



Each Ball:

- `GOval`
- `ballVx`
- `ballVy`

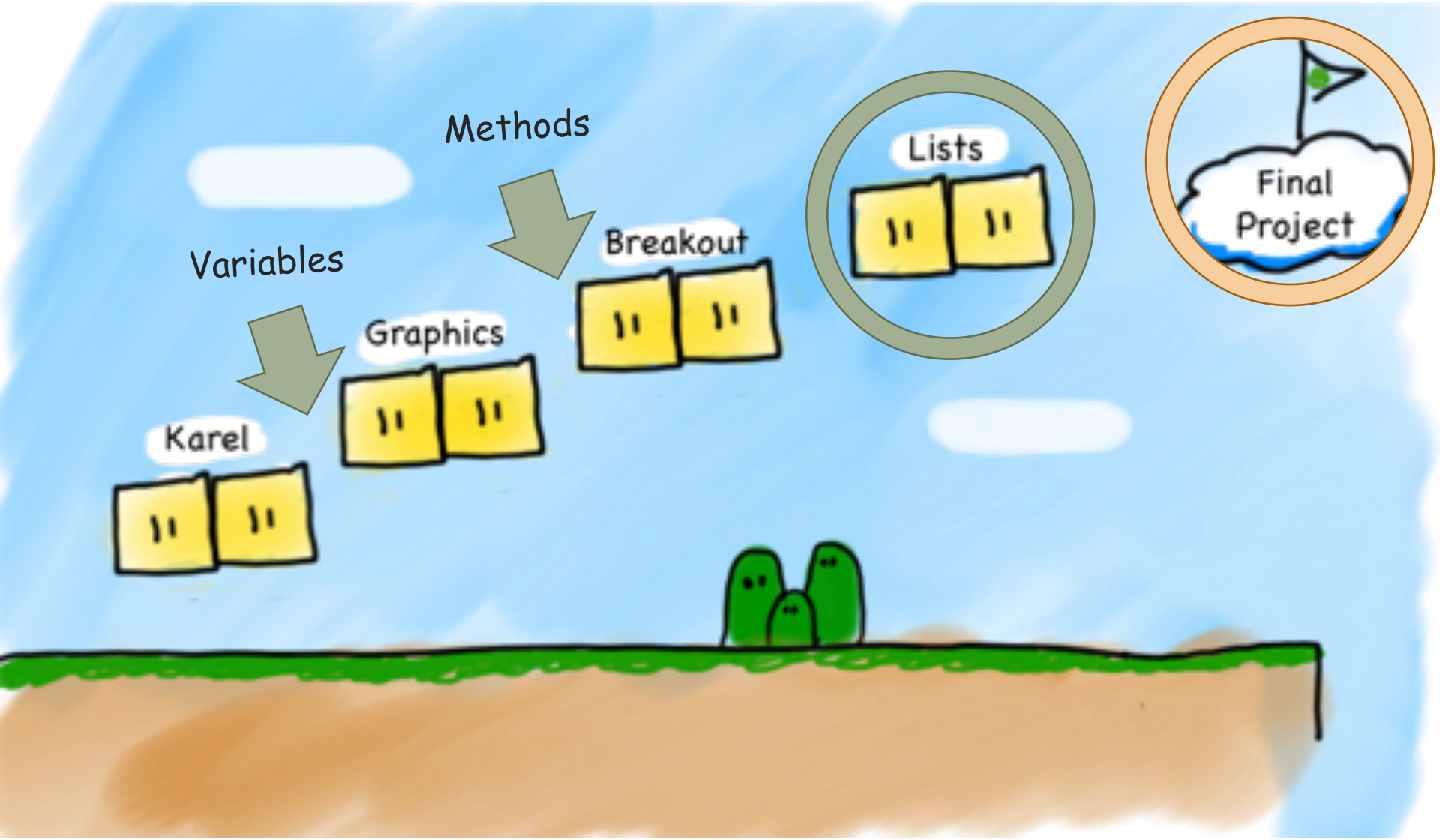
(1) Setup



(2) Animate

```
// for the i-th ball
GOval ball = balls.get(i);

// update i-th vx/vy
// (perform wall bounce)

// move ball
ball.move(ballsVx.get(i),
          ballsVx.get(i));
```

Our Last Step



Your Final Project is like İskender

Excellent, existing ideas

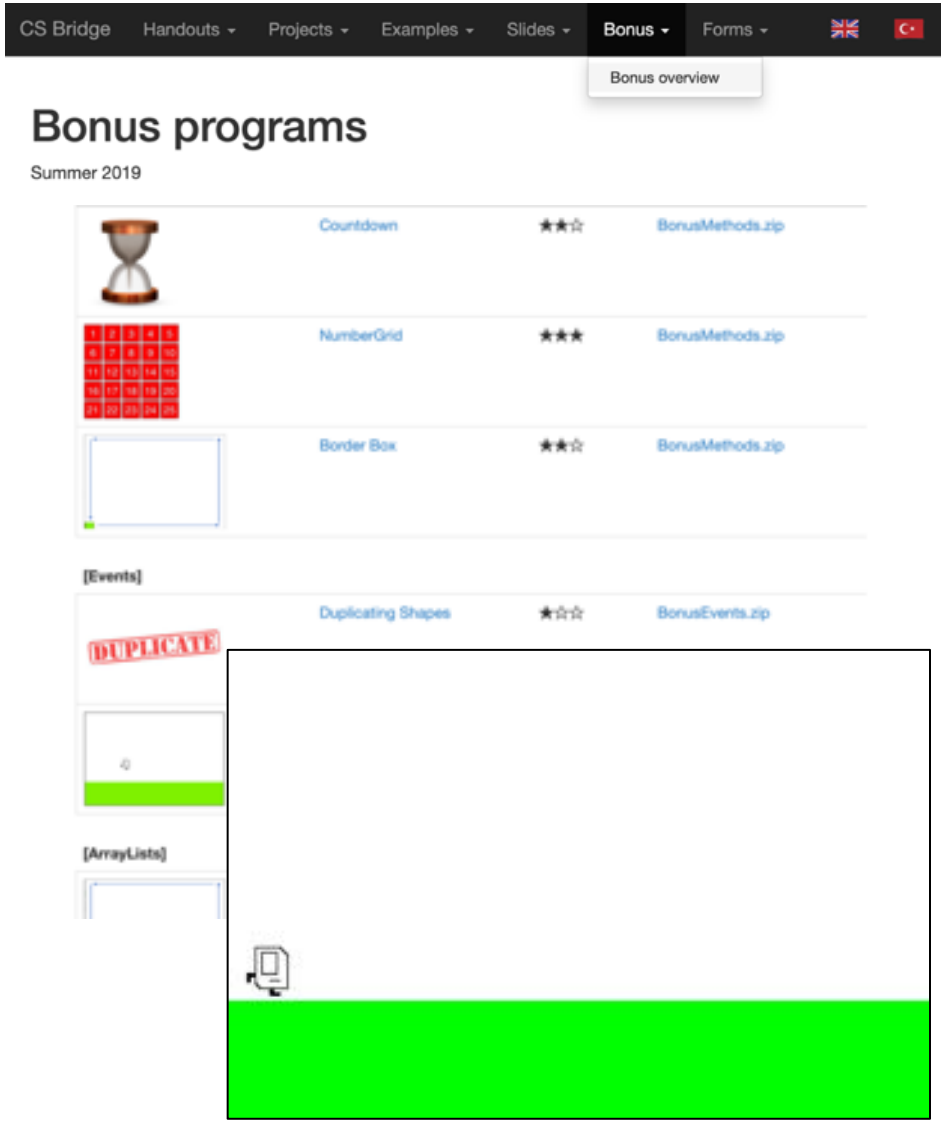
Your projects, worked examples

Some basics

Think outside the box

Lots of Help


Projects
Examples
Slid
Step Up
Place 100
Beeper Line
Invert
UN Karel
E=MC2
Fibonacci
Find Pi
8-Ball
Average Method
Robot Face
Draw People
Half Green
Go To Center
Gravity Ball
Stamp Tool
Hole Puncher
Debris Sweeper
Such a Drag
Keyboard Karel
Racing Cars
Anatolian Rock
Rocket Paddle
Bouncing Balls
Name
Min Max Me


CS Bridge
Handouts
Projects
Examples
Slides
Bonus
Forms
Bonus overview
Bonus programs
Summer 2019
Countdown
BonusMethods.zip
NumberGrid
BonusMethods.zip
Border Box
BonusMethods.zip
[Events]
Duplicating Shapes
BonusEvents.zip
DUPLICATE
[ArrayLists]

Lots of Help


Lisa


Barış


Kaan


Ahmet


Beyzanur


Ceren


Ece


Eren


Ezgi


Gül Sena


Haluk


Hasan


İpek


Levent


Necla


Oğuzhan


Ozan D


Ozan N


Quincy


Sabri


Seher


Serhat

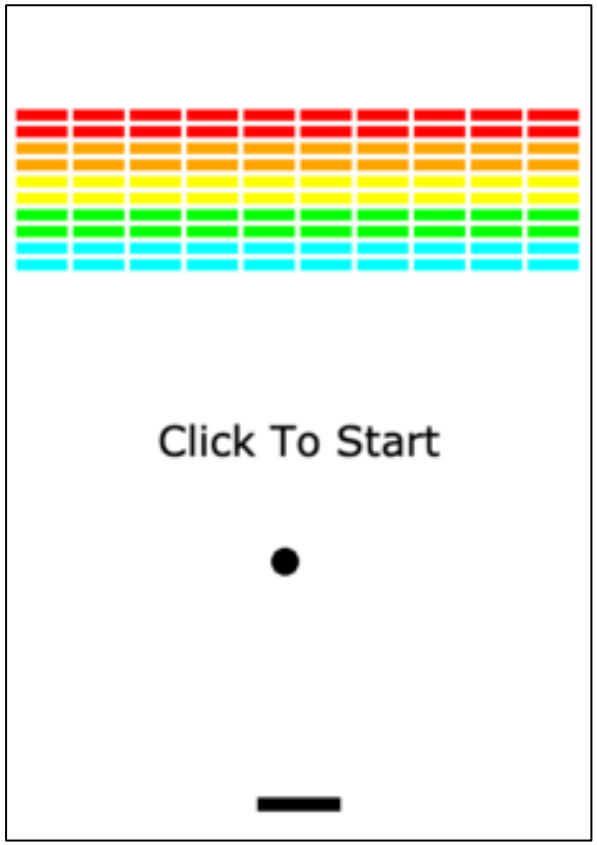

Ayça


Chris


Nick


Asena

Your goals today



(1) Breakout: Finish it up!

(2) Array exercise: MinMaxMean
(+ ArrayList exercises)
for tomorrow

(3) Get Final Project idea approved

Console/Graphics,
Games/Stories,
Puzzles/Adventures,
Math/Medicine/Science,
…The ArrayList goes on!