CSSE 220 Day 5

Iteration and Debugging
Arrays and ArrayLists

Check out *Iteration* and *ArraysAndLists* from SVN

Questions?

Arc2D Example

- If you had trouble with Arc2D on Faces, take a look at ArcExample.java and ArcDrawer.java after class.

Review Loops: while & for Loops

- While loop syntax: **Same as C**

```
while (condition) {
  statements
}
```

- For loop syntax: **Similar to C**

```
for (initialization ; condition ; update) {
    statements
}
```

In both cases, curly braces optional if only one statement in body; but be careful!

Java Loop Examples

- Look at `Investment.java`, `InvestmentTest.java` and `InvestmentRunner.java`
  - Practice using a single **while** loop
  - Study and run the code, then answer quiz questions

- Do the **Rates** exercise in today's homework
  - You'll practice using a single **for** loop in that exercise
  - **Hint**: in printf's format string, use %% to display a single %

- If you finish the **Rates** exercise, start on the **Pyramid Scheme** exercise described in homework
  - You'll practice **nested loops** in that exercise

Q1 – Q2

Sentinel Values: A Loop and a Half

- *Sentinel value*—a special input value not part of the data, used to indicate end of data set
  - `Enter a quiz score, or Q to quit:`

- *A loop and a half*—a loop where the test for termination comes in the **middle** of the loop

- Examples… (on next slide)

Q3-Q4

Two Loop-and-a-half Patterns

```
// Pattern 1

boolean done = false;
while (!done) {
   // do some work

   if (condition) {
      done = true;
   } else {
      // do more work
   }
}
```

The variable *done* here is called a *flag*

```
// Pattern 2

while (true) {
   // do some work

   if (condition) {
      break;
   }

   // do more work
}
```

Q5

Debugging—Key Concepts

- Breakpoint
- Single stepping
- Inspecting variables

Q6

Debugging—Demo

- Debugging Java programs in Eclipse:
  - Launch using the debugger
  - Setting a breakpoint
  - Single stepping: *step over* and *step into*
  - Inspecting variables
- Complete **WhackABug** exercise

Q7

Array Types

- Group a collection of objects under a single name
- Elements are referred to by their **position**, or ***index***, in the collection (0, 1, 2, …)
- Syntax for declaring: *ElementType*`[]` *name*
- Declaration examples:
  - A local variable: `double[ ] averages;`
  - Parameters: `public int max(int[] values) {…}`
  - A field: `private Investment[] mutualFunds;`

Allocating Arrays

- Syntax for allocating:

  `new ElementType[length]`

- Creates space to hold values
- Sets values to defaults
  - `0` for number types
  - `false` for boolean type
  - `null` for object types
- Examples:
  - `double[] polls = new double[50];`
  - `int[] elecVotes = new int[50];`
  - `Dog[] dogs = new Dog[50];`

Don't forget this step!

This does NOT construct any **Dog**s. It just allocates space for referring to **Dog**s (all the **Dog**s start out as *null*)

Q8

Reading and Writing Array Elements

- Reading:
  - `double exp = polls[42] * elecVotes[42];`

Reads the element with index 42.

Sets the value in slot 37.

- Writing:
  - `elecVotes[37] = 11;`

- Index numbers run from 0 to array length – 1
- Getting array length: `elecVotes.length`

No parentheses, array length is (like) a field

Q9–Q10

Arrays: Comparison Shopping

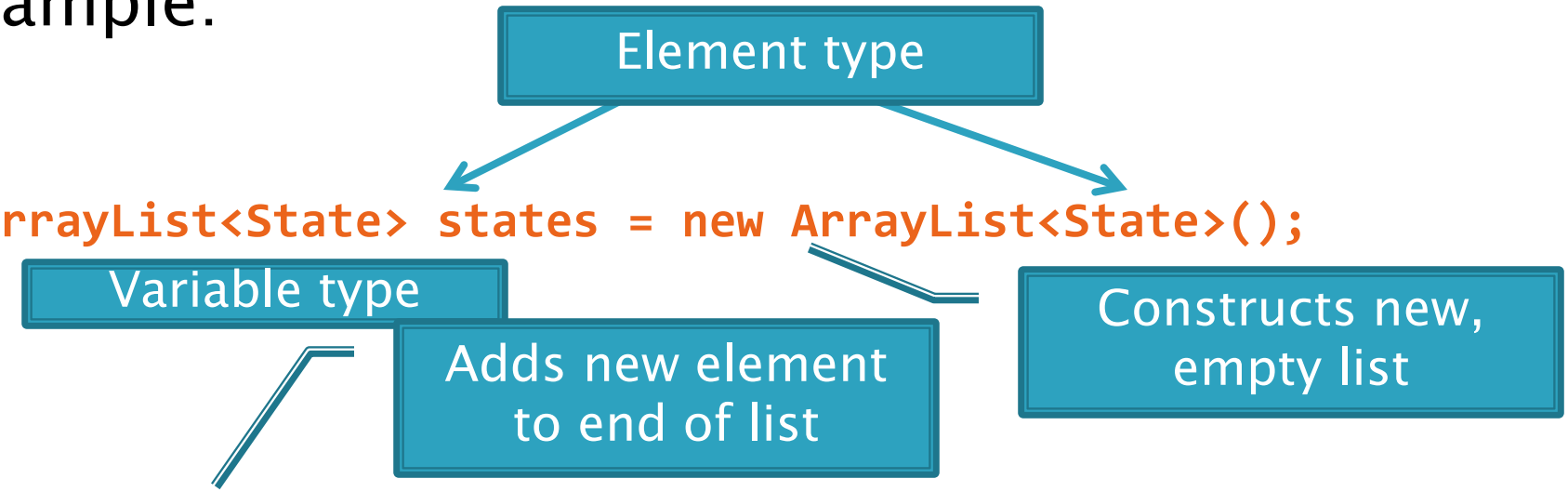
Arrays…	Java	C	Python lists
have fixed length	yes	yes	no
are initialized to default values	yes	no	n/a
track their own length	yes	no	yes
trying to access “out of bounds” stops program before worse things happen	yes	no	yes

Live Coding

- Investigating the Law of Large Numbers
  - A simulation using dice
- Design
- Implementation (together)
- Begin the **RollingDice** program for HW5b (in **ArraysAndLists project**)

What if we don't know how many elements there will be?

- **ArrayLists** to the rescue
- Example:



- `ArrayList` is a *generic class*
  - Type in <brackets> is called a *type parameter*

Q11-Q12

ArrayList Gotchas

- Type parameter can't be a primitive type
  - Not: `ArrayList<int> runs;`
  - But: `ArrayList<Integer> runs;`

- Use *get* method to read elements
  - Not: `runs[12]`
  - But: `runs.get(12)`

- Use **`size()`** not **`length`**
  - Not: `runs.length`
  - But: `runs.size()`

Lots of Ways to Add to List

- Add to end:
  - `victories.add(new WorldSeries(2011));`
- Overwrite existing element:
  - `victories.set(0,new WorldSeries(1907));`
- Insert in the middle:
  - `victories.add(1, new WorldSeries(1908));`
  - Pushes elements at indexes 1 and higher up one

- Can also remove:
  - `victories.remove(victories.size() - 1)`

Live Coding

- Continue **RollingDice**

Work Time

- Wrap up **Rates and PyramidScheme** if you haven't already, then continue working on homework

Q13–Q14, turn in quiz now