CSSE 220

Arrays, ArrayLists,
Wrapper Classes, Auto-boxing,
Enhanced *for* loop

Check out *ArraysListPractice* from SVN

Syllabus Highlights

- You should read the whole thing
- But pay special attention to the grading policies of the course

Review of types

- Primitives
  - int, double, char, boolean, long, …
- Objects
  - String, …
- Gotchas:

  What is 7/2?

  Alternatives?

  What is x/y if x and y are both ints?

  Alternatives?

  What is s after these 2 lines?

```
String s = “computer”;
s.substring(0,3);
```

  Alternatives?

Arrays- What, When, Why, & How?

- What
  - A special **type** used to hold a set number of items of a specified type
- When
  - Use when you need to store multiple items of the same type
  - Number of items is known and **will not change**

Arrays- What, When, Why, & How?

- Why
  - Avoids things like int1, int2, int3, int4
  - Avoids repetitive code and frequent updates
- How
  - Type[] arr = new Type[num]; ← Creates a new array of type Type stored in variable arr
  - An array of 5 Strings (stored in the variable fiveStrings) would look like this:
    - String[] fiveStrings = new String[5];

Array Examples Handout

- Form groups of 2
- Look at the Array Examples Handout
- Study how arrays are used and answer the questions in the quiz
  - FIRST PAGE OF QUIZ ONLY

Go to http://codingbat.com/java/Array-2

- Work in your groups to solve fizArray3, bigDiff, shiftLeft
- If you finish early, try zeroFront

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*)

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

Arrays: Comparison Shopping

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

ArrayList- What, When, Why, & How?

- What
  - A class in a Java library used to hold a collection of items of a specified type
  - Allows variable number of items
  - Fast random access
- When
  - Use when you need to store multiple items of the same type
  - Number of items is not known/will change

ArrayList- What, When, Why, & How?

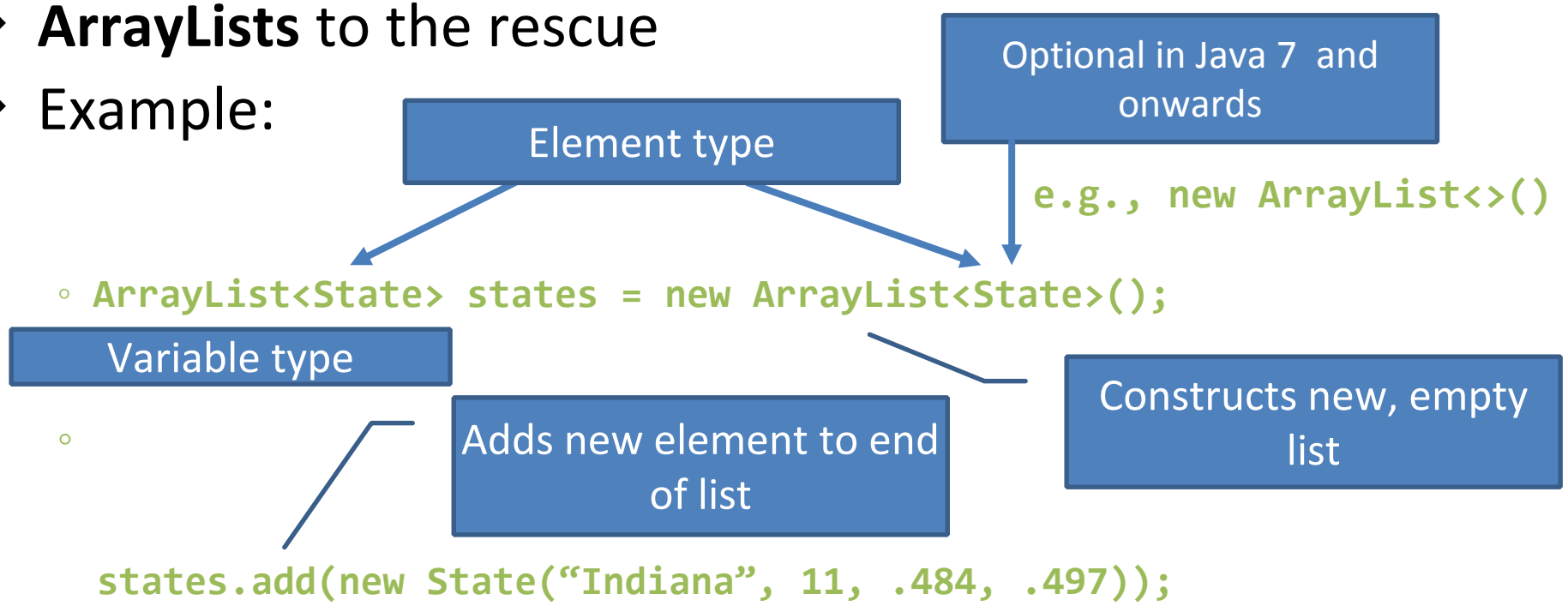
- Why
  - Fast random access
  - Allows length changes, cannot do this with an array
- How
  - `ArrayList<Type> arl = new ArrayList<Type>();`
    - Creates a new ArrayList of type Type stored in variable arl

ArrayList Examples Handout

- Look at the ArrayList section of the examples handout
- Study how arrayLists are used and answer the questions in the quiz
- Then solve the 3 problems in ArrayListPractice (you downloaded it from SVN)

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*

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)`

So, what's the deal with primitive types?

- Problem:
  - ArrayList's only hold objects
  - Primitive types aren't objects

- Solution:
  - *Wrapper classes*—instances are used to "turn" primitive types into objects
  - Primitive value is stored in a field inside the object

Primitive	Wrapper
byte	***Byte***
boolean	***Boolean***
char	***Character***
double	***Double***
float	***Float***
int	***Integer***
long	***Long***
short	***Short***

Work Time

- Finish all the in-class material exercises if you haven’t yet
- Work on TwelveProblems