ArrayList

CPSC 1181 – O.O.

Jeremy Hilliker

Summer 2017

Outline

- ArrayList
- Wrappers
- Auto-Boxing
- Enhanced for Loop

Problems with Arrays

- Their size is fixed
- What happens if you need it to be bigger?
- What if you don’t end up filling it?

ArrayList

- A list (read: ordered sequence) of **Objects**
  - Implemented with an array
    - Not that that should matter
  - http://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

- Major methods:
  - **size**
  - **add**
  - **contains**
  - **get**
  - **indexOf**
  - **isEmpty**
  - **iterator**
  - **remove**
  - **set**
- Sometimes used:
  - clear
  - forEach
  - removeIf
  - subList
  - toArray
  - trimToSize

ArrayList<T>

- `ArrayList` is a "generic" class
  - a/k/a a "template type"
  - **`ArrayList<T>`** holds objects of type **`T`**
  - Read as: "ArrayList of T"

```
4       ArrayList<String> names = new ArrayList<String>();
5       names.add("Alice");
6       names.add("Bob");
7       names.add("Claire");
8       System.out.println(names.size());   // 3
```

Getting Elements

- Same sort of rules as an array
  - Cannot exceed the bounds

```
4       ArrayList<String> names = new ArrayList<String>();
5       names.add("Alice");
6       names.add("Bob");
7       names.add("Claire");
8       System.out.println(names.size());   // 3
9
10      i = names.size(); // 3
11      String aName = names.get(0);  // "Alice"
12      String bName = names.get(names.size()); // error
13      // legal values are 0 ... size-1
```

Adding Elements

- add(E)
  - Adds to the end of the list
- add(index, E)
  - Inserts at specified position
  - Shifts subsequent elements up one position
- set(index, E)
  - Replaces the element at the specified position

Removing Elements

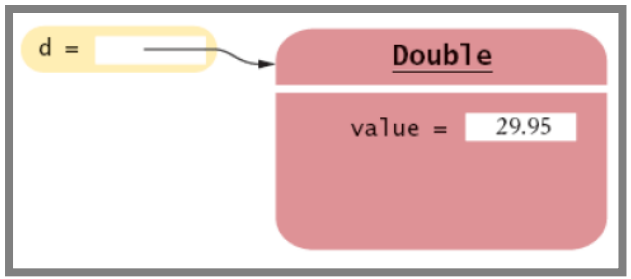
- remove(index)
  - Removes the element at the specified position
  - Shifts subsequent elements down one position

- remove(Object o)
  - Removes the first occurrence of the element that is .equals (or null & null)
  - Shifts subsequent elements down

Wrappers

- ArrayList holds objects, not primitive types
- What if we want to hold doubles? (or others)
- There exist wrapper classes:

```
15  ArrayList<Double> ds = new ArrayList<Double>();
16  ds.add(29.95); // ds.add(new Double(29.95))
17  Double d = ds.get(0);
18  double primitive = d;   // d.doubleValue();
```



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

Auto-Boxing

- As of java 5.0, conversion between primitive and wrapper is automatic

```
15   Double d = 29.95; // auto-boxed
16   // ==
17   Double d = new Double(29.95);
18
19   double x = d;  // auto-unboxed
20   // ==
21   double x = d.doubleValue();
22
23   // previous slide ==
24   ArrayList<Double> ds = new ArrayList<Double>();
25   ds.add(29.95);     // auto-unboxed
26   double primative = ds.get(0); // auto-unboxed
```

Enhanced for Loop

```
28  Random r = new Random();
29  ArrayList<Double> ds = new ArrayList<double>();
30  for(int i = 0; i < 16; i++) {
31      ds.add(r.nextDouble()); // double
32  }
33  for(double d : ds) {
34      System.out.println(d);
35  }
```

Examples:

- Counting matches
- Finding a value

Outline

- ArrayList
  - Enhanced for Loop
- Wrappers
  - Auto-Boxing