CS159

Nathan Sprague

November 11, 2013

Solving the Problem of Fixed Length Arrays

DynamicArray.java ↗ DynamicArrayDriver.java ↗

ArrayList

Naming convention for Java Collection types: ArrayList

- Array - Coded using arrays "under the hood".
- List - Implements the List interface ↗.

- ArrayList API ↗

Collections

- Collection - a class that stores multiple elements.
- We will distinguish between:
    - The interface to a collection - how we interact with the collection.
    - The implementation of the collection - how the data is stored "behind the scenes".

- Java Collections Overview ↗
- Java Collections Interfaces Overview ↗

Java Arrays

- Note that Java Arrays are in a category by themselves:
  - Not quite objects, not quite primitive types.
  - An array is NOT an object of type array
    - Has no methods.
    - cannot be subclassed.
    - *does* have fields: `myArray.length`

- Advantages:
  - efficient.
  - familiar(?) syntax borrowed from other languages.
- Disadvantages:
  - Fixed length.
  - Awkwardly different from all other collections.

Question

```
1  ArrayList < Integer > nums = new ArrayList < Integer >();
2  nums . add (150);
3  nums . add (200);
4  System . out . println ( nums . get (1). toString ());
```

1 Does not compile.

2 Compiles, but throws an exception at run time.

3 Runs without error.

Autoboxing

- What is going on here?
    - Java automatically converts primitive types to reference types when necessary.
    - `nums.add(150);`
    - silently becomes:
    - `nums.add(new Integer(150));`

```
1  whichCourse["Nathan"] = "CS159"
2  System.out.println(whichCourse["Nathan"]);
```

1. Does not compile.
2. Compiles, but throws an exception at run time.
3. Runs without error.

(Assuming `whichCourse` is properly initialized.)

Question

```
1  whichCourse["Nathan"] = "CS159"
2  System.out.println(whichCourse["Nathan"]);
```

1. Does not compile.
2. Compiles, but throws an exception at run time.
3. Runs without error.

(Assuming whichCourse is properly initialized.)

- Too bad. This would be handy.

HashMap

Recall the Naming Convention: HashMap

- Map - Implements the Map interface ↗.
    - A Map maps from a "key" object to a "value" object.
    - Also called a Dictionary or Associative Array.

- Hash - Coded using a hash table (Something to look forward to in CS240!)

- HashMap API ↗

Example:
HashMapDemo.java ↗

Iterators

- Iterators provide a common mechanism for iterating through Java Collections.
- An iterator is an object that implements the Iterator Interface↗.

Example:
IteratorDemo.java↗

Iterable

- Most Java Collection types implement the Iterable interface↗.
- This is the magic sauce behind for-each loops.

```
1  for (String s : someCollection)
2      System.out.println(s);
```

Is (pretty much) just a shorthand for:

```
1  Iterator<String> it = SomeCollection.iterator();
2  String s;
3  while(it.hasNext())
4  {
5    s = it.getNext();
6    System.out.println(s);
7  }
```

Question

```
1          String[] strings = new String[2];
2          strings[0] = "hello";
3          strings[1] = "bob";
4
5          for (String s : strings)
6              System.out.println(s);
```

1 Does not compile.

2 Compiles, but throws an exception at run time.

3 Runs without error.

Question

```java
public static void main(String[] args)
{
    String[] strings = new String[2];
    strings[0] = "hello";
    strings[1] = "bob";
    printCollection(strings);
}

public static void printCollection(Iterable collection)
{
    for (Object o : collection)
    {
        System.out.println(o);
    }
}
```

1 Does not compile.
2 Compiles, but throws an exception at run time.
3 Runs without error.