COMP 110/L Lecture 16

Mahdi Ebrahimi

Slides adapted from Dr. Kyle Dewey

Outline

- Looping over multiple arrays
- Testing with arrays

break vs continue

You have already seen the break statement used to "jump out" of a switch statement.

The **break** statement can also be used to jump out of a loop.

The **continue** statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

Example

For-Each Loop

There is also a "**for-each**" loop, which is used exclusively to loop through elements in an **array**:

```
for (type variableName: arrayName) {
    // code block to be executed
}
```

Example:

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (String car: cars) {
    System.out.println(car);
}
```

Looping Over Multiple Arrays

Same index variable can be used on multiple arrays

Example: `LoopTwo.java`

Creating Arrays from Arrays

Similar pattern arises if trying to make an array from an array

Examples:

`CopyArray.java`

`CopyFirstThree.java`

Testing with Arrays

JUnit Recap

You've been using JUnit's `assertEquals` for awhile...

JUnit Recap

You've been using JUnit's `assertEquals` for awhile...

```
import static org.junit.Assert.assertEquals;
```

JUnit Recap

You've been using JUnit's `assertEquals` for awhile...

```
import static org.junit.Assert.assertEquals;


@Test
public void testSomething() {
  assertEquals(2, MyMethod.myMethod(1));
}
```

JUnit with Arrays

Can use `assertArrayEquals` to look at array contents

JUnit with Arrays

Can use `assertArrayEquals` to look at array contents

```
import static org.junit.Assert.assertArrayEquals;
```

JUnit with Arrays

Can use `assertArrayEquals` to look at array contents

```
import static org.junit.Assert.assertArrayEquals;


 @Test
 public void testSomething() {
    assertArrayEquals(new int[]{1, 2, 3},
                      MyMethod.myMethod(1));
 }
```

Example

- `ParseStrings.java`
- `ParseStringsTest.java`

Writing Tests for Loops over Arrays

- Which tests are interesting tends to be problem-specific
- Often of value: arrays of length 0, 1, and 2