Parameter Lists & Command Line Arguments

October 10, 2007

Example: Numerical Integration

Quick review of last lecture

Arrays

• Another way to depict the scores array:

Arrays of Objects

• The words array when initially declared:

• At this point, the following reference would throw a NullPointerException:
  System.out.println (words[0]);

Arrays of Objects

• After some string objects are created and stored in the array:
Initial Values for Arrays of Objects

• Keep in mind that String objects can be created using literals
• The following declaration creates an array object called verbs and fills it with four String objects created using string literals

```java
String[] verbs = {"play", "work", "eat", "sleep"};
```

Another Way to do the Same Thing

• Initialize each element separately

```java
String[] verbs;
verbs = new String[4];
verbs[0] = new String("play");
verbs[1] = new String("work");
verbs[3] = new String("sleep");
```

CD Collection Example

• Now let’s look at an example that manages a collection of CD objects
• See Tunes.java (page 387)
• See CDCollection.java (page 388)
• See CD.java (page 391)

Other Stuff

Command-Line Arguments

• The signature of the main method indicates that it takes an array of String objects as a parameter
• These values come from command-line arguments that are provided when the interpreter is invoked
• For example, the following invocation of the interpreter passes three String objects into main:
  ```shell
  java StateEval pennsylvania texas arizona
  ```
• These strings are stored at indexes 0-2 of the array parameter of the main method
• See NameTag.java (page 393)
Variable Length Parameter Lists

• Suppose we wanted to create a method that processed a different amount of data from one invocation to the next
• For example, let’s define a method called average that returns the average of a set of integer parameters

  // one call to average three values
  mean1 = average (42, 69, 37);

  // another call to average seven values
  mean2 = average (35, 43, 93, 23, 40, 21, 75);

Variable Length Parameter Lists

• We could define overloaded versions of the average method
  • Downside: we’d need a separate version of the method for each parameter count
• We could define the method to accept an array of integers
  • Downside: we’d have to create the array and store the integers prior to calling the method each time
• Instead, Java provides a convenient way to create variable length parameter lists

public double average (int ... list)
{
  double result = 0.0;
  if (list.length != 0)
  {
    int sum = 0;
    for (int num : list)
      sum += num;
    result = (double)sum / list.length;
  }
  return result;
}
Variable Length Parameter Lists

- The type of the parameter can be any primitive or object type

```java
public void printGrades (Grade ... grades)
{
    for (Grade letterGrade : grades)
        System.out.println (letterGrade);
}
```

- A method that accepts a variable number of parameters can also accept other parameters

```java
public void test (int count, String name,
                double ... nums)
{
    // whatever
}
```

- The varying number of parameters must come last in the formal arguments
- A single method cannot accept two sets of varying parameters
- Constructors can also be set up to accept a variable number of parameters

Example: `VariableParameters.java` (page 396)

Example: `Family.java` (page 397)

THE END