415.101

Java

Christian Collberg
August 16, 1997

Tutorial #5

Copyright © 1997 C. Collberg

This Week’s Tutorial

- for-loops
- do-loops
- while-loops
- The StringTokenizer class.
- One-dimensional arrays.
- We will use a running example to illustrate these concepts: Computing and displaying scores in an athletic competition.

Slide 5–1

For-Loops I

- Use a for-loop when you know how many times you want to execute the loop body when the loop is entered.
- Structure:
  ```java
  for (start; check; update) {
      body
  }
  ```
- start declares and initializes a loop variable.
- check is a condition that determines when the loop should end.
- update gives a new value to the loop variable each time around the loop.
- The loop variable is most often an integer.

Slide 5–2

For-Loops II

Examples:

```
for(int i=1; i<=10; i++)
    System.out.println("Hello World!");
```

```
for(int i=10; i>=1; i--)
    System.out.println("Countdown: "+i);
```
The Event Program

- Write a program which reads in the names of two tri-athletes and their three individual scores.
- The program should compute the total score for each contestant and print the name of the winner, if any.
- Use for-loops to read the three individual scores!

```java
class Event {
    public static void main(String[] args) {
        String FirstContestantScore = 0;
        for (int i = 0; i < 3; i++) {
            System.out.println("Enter the 1st contestant's name:");
            String FirstContestant = Input.toText("Enter the 1st contestant's name:");
            System.out.println("Enter score number "+i+1+":");
            FirstContestantScore += Input.toInt(""+FirstContestantScore);
        }
        System.out.println("Lisa wins!");
    }
}
```

Arrays I

- We use arrays to store collections of the same kind of data item
- Declaring an initialized array:
  ```java
type name [] = {values};
```
- Declaring an uninitialized array:
  ```java
type name [] = new type [limit];
```
- The array elements are accessed through indexing. The first element of an array A is in A[0], the second in A[1], the last in A[A.length-1].
- Examples:
  ```java
  String colors [] = { "red", "blue" };  
  double heights [] = new double [10];  
  heights[5] = 5.7;  
  System.out.println(colors[1]);  
  System.out.println(heights[5]);
  ```

Arrays II

- Arrays are indexed starting at 0.
- It is an error to index outside the bounds of an array:
  ```java
double H [] = new double [10];  
H[0] = 5.7;  // OK!  
H[9] = 5.7;  // OK!  
System.out.println(H[10]);  // Wrong!  
H[-1] = 5.7;  // Wrong!  
System.out.println(H[10]);  // Wrong!
```
Event: Initialized Arrays

- Write a program which reads in the names of two tri-athletes and their three individual scores.
- The program should compute the total score for each contestant and print the name of the winner, if any.
- Use an initialized array to allow the program to print out more specific prompts (1st, 2nd, 3rd!)

```java
Event:
public static void main(String[] args) {
    String FirstContestantScore = 0;
    for (int i = 0; i < FirstContestantScore; i++) {
        // As before!
    }
}
```

Slide 5–8

Event: For-loops & Arrays

- Write a program which reads in the names of two tri-athletes and their three individual scores.
- The program should print a table with the contestants’ names and total scores.
- Use for-loops to read the contestants’ names and their individual scores! Use arrays to store the names and scores!

```java
public static void main(String[] args) {
    String FirstContestantScore = 0;
    for (int i = 0; i < FirstContestantScore; i++) {
        // As before!
    }
}
```

Slide 5–9

```java
public static void main(String[] args) {
    String FirstContestantScore = 0;
    for (int i = 0; i < FirstContestantScore; i++) {
        // As before!
    }
}
```

Slide 5–10

```java
public static void main(String[] args) {
    String FirstContestantScore = 0;
    for (int i = 0; i < FirstContestantScore; i++) {
        // As before!
    }
}
```

Slide 5–11
Use the `StringTokenizer` class to break up a string entered by the user (e.g., read using `Input.toText`).

You must import `java.util.*`.

Create a new tokenizer by calling `StringTokenizer(S)` on the string.

Call `nextToken()` to get the next part of the string.

Call `hasMoreTokens()` to see if there are any tokens left in the string.

```java
import java.util.*;

String S = Input.toText();
StringTokenizer T = new StringTokenizer(S);
String Text = T.nextToken();
int value = Integer.valueOf(Text.trim()).intValue();
while (T.hasMoreTokens())
    String V = T.nextToken();
```

---

**Event: StringTokenizer I**

Write a program which reads in the names of any number of tri-athletes and their three individual scores and prints a table with the contestants’ names and total scores.

Use `StringTokenizer` to read the contestants’ names and their individual scores.

```java
public class StringPartitioner {
    public static void main(String[] args) {
        StringTokenizer T = new StringTokenizer(S);
        String name; int score; int sum = 0;
        for (int i = 0; T.hasMoreTokens(); i++) {
            name = T.nextToken();
            StringTokenizer S = new StringTokenizer(T.nextToken());
            for (int j = 0; S.hasMoreTokens(); j++) {
                score = Integer.valueOf(S.nextToken()).intValue();
                sum += score;
            }
            System.out.println(name + " Score: " + sum);
            sum = 0;
        }
    }
}
```
Do and While Loops I

- Use a do-loop when you know that the loop body should be executed at least once.
- Use a while-loop when the loop body could be executed zero or more times.
- Structure:
  initialize condition
  while (condition) {
    statements
    update condition
  }
  initialize condition
  do {
    statements
  } while (condition)

Do and While Loops II

Examples:

```java
int i = Input.toInt("");  
while (i < 100) {
    System.out.println(i);
    i = Input.toInt("");  
}
```

```java
int i;
   do {
      i = Input.toInt("");
      System.out.println(i);
   } while (i < 100)
```

- Sometimes an indeterminate loop with a jump out from the middle makes for simpler loop conditions:

```java
while (true) {   // or 'for(;;')
    if (some condition) break;
}
```

---

Event: StringTokenizer II

- Write a program which reads in the names of any number of athletes and any number of individual scores and prints a table with the contestants' names and total scores.
- Don't prompt for the number of athletes. Rather, let the user terminate the input by typing 'END'!
- Use hasMoreTokens() to test for the last score!

```java
class Event {  
    public static int NumberOfContestants = 10;  
    public static void main(String[] args) {  
        int Contests = 0;  
        boolean Done = false;
        String S = Input.printf(Placeholder String, getopt());  
        do {  
            String Name = getopt();  
            String Score = getopt();  
            int NewScore = Integer.parseInt(S + getopt());  
            if (Done) break;  
            if (Name.charAt(0) == 'C') {  
                Score[Contests] = NewScore;
                Contests++;
            }  
            if (Score[Contests] == 0) 
                break;  
        } while (true);  
    }
}
```
This time, check that the user doesn't enter too many athletes. We must not index outside the array bounds!

This time, don't use a boolean `Done` variable. Rather, use an infinite loop with a `break` in the middle:

```java
while (true) {
    // some code here
    if (some condition) break;
    // some code here
}
```

---

**Event: Histogram**

This time, print out a horizontal histogram of the results.

Also compute and print the average score.

Make sure not to print too long lines! A typical window may have a width of 60–80 characters.
- This time, print out a **vertical** histogram of the results.
- The program is now **59** lines long!