University of Auckland

415.101

Java

Christian Collberg

August 16, 1997

Tutorial #5

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:

```
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:

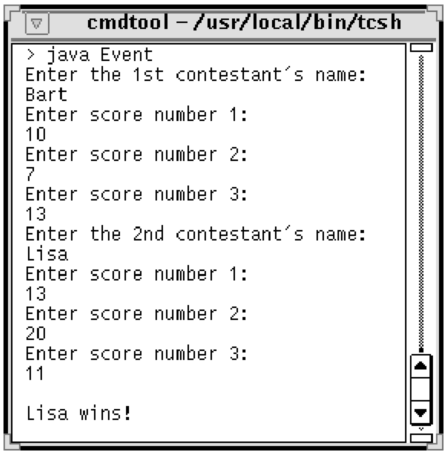
```
for(int i=1; i<=10; i++)
   System.out.println("Hello World!");

for(int i=10; i<=1; i--)
   System.out.println("Countdown: "+i);
```

Slide 5–3

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!



Slide 5–4

```
class Event {
  public static void main (String [] args) {
    String FirstContestant = Input.toText("Enter the 1st contestant's name:");

    int FirstContestantScore  = 0;
    for (int i=1; i<=3; i++)
      FirstContestantScore  += Input.toInt ("Enter score number " + i + ": ");

    String SecondContestant = Input.toText ("Enter the 2nd contestant's name:");
    int SecondContestantScore  = 0;
    for (int i=1; i<=3; i++)
      SecondContestantScore  += Input.toInt ("Enter score number " + i + ": ");

    String Winner = "Noone";
    if (FirstContestantScore > SecondContestantScore)
      Winner = FirstContestant;
    else if (FirstContestantScore < SecondContestantScore)
      Winner = SecondContestant;
    System.out.println("\n" + Winner + " wins!");
  }
}
```

Slide 5–5

Arrays I

- We use arrays to store collections of
  the same kind of data item
- Declaring an initialized array:

  `type name [] = { values };`
- Declaring an uninitialized array:

  `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:

```
String colors [] = { "red", "blue" };
double heights [] = new double [10];
heights[5] = 5.7;
System.out.println(colors[1]);
System.out.println(heights[5]);
```

Slide 5–6

Arrays II

- Arrays are indexed starting at 0.
- It is an error to index outside the bounds of an array:

```
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!
```

Slide 5–7

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)!

```
cmdtool - /usr/local/bin/tcsh
> java EventA
Enter the 1st contestant's name:
Bart
Enter Bart's 1st score:
5
Enter Bart's 2nd score:
10
Enter Bart's 3rd score:
15
Enter the 2nd contestant's name:
Lisa
Enter Lisa's 1st score:
6
Enter Lisa's 2nd score:
11
Enter Lisa's 3rd score:
16

Lisa wins!
>
```

Slide 5–8

```
class EventA {
  public static void main (String [] args) {

     String OrderNames [] = {"1st", "2nd", "3rd"};

     String FirstContestant = Input.toText("Enter the 1st contestant's name:");

     int FirstContestantScore  = 0;
     for (int i=0; i<3; i++)
        FirstContestantScore  += Input.toInt (
           "Enter " + FirstContestant + "'s " + OrderNames[i] + " score: ");

     String SecondContestant = Input.toText ("Enter the 2nd [...] name:");
     int SecondContestantScore  = 0;
     for (int i=0; i<3; i++)
        SecondContestantScore  += Input.toInt (
           "Enter " + SecondContestant + "'s " + OrderNames[i] + " score: ");

     // As before!
  }
}
```

Slide 5–9

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!

```
cmdtool - /usr/local/bin/tcsh
> java EventB
Enter the 1st contestant's name:
Bart
Enter Bart's 1st score:
2
Enter Bart's 2nd score:
3
Enter Bart's 3rd score:
4
Enter the 2nd contestant's name:
Lisa
Enter Lisa's 1st score:
5
Enter Lisa's 2nd score:
6
Enter Lisa's 3rd score:
7


Name            Score
----------------------------------
Bart            9
Lisa            18
>
```

Slide 5–10

```
class EventB {
  public static int NoOfContestants = 2;
  public static void main (String [] args) {
     String OrderNames [] = {"1st", "2nd", "3rd"};
     String Name  [] = new String [NoOfContestants];
     int    Score [] = new int    [NoOfContestants];

     for (int c=0; c<NoOfContestants; c++) {
        Name[c]  = Input.toText("Enter the " + OrderNames[c] +
                                " contestant's name:");
        Score[c] = 0;
        for (int i=0; i<3; i++)
           Score[c] += Input.toInt (
             "Enter " + Name[c] + "'s " + OrderNames[i] + " score: ");
     }

     System.out.println("\n\nName\t\tScore");
     System.out.println("----------------------------------");
     for (int c=0; c<NoOfContestants; c++)
        System.out.println(Name[c] + "\t\t" + Score[c]);
}}
```

Slide 5–11

StringTokenizer I

- 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.

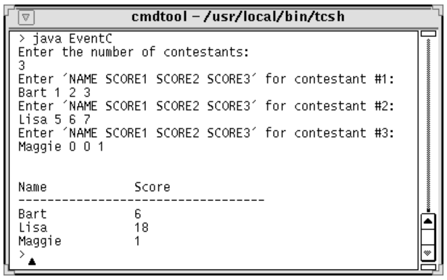
```
import java.util.*;
   ....
String S = Input.toText();
StringTokenizer T = new StringTokenizer(S);
String Text = T.nextToken();
int Value=Integer.valueOf(V.trim()).intValue();
while (T.hasMoreTokens())
    String V = T.nextToken();
```


StringTokenizer II

```
public class StringTokenizer implements Enumeration {
   public StringTokenizer(String str, String delim,
                  boolean returnTokens);
   public StringTokenizer(String str, String delim);
   public StringTokenizer(String str);
   public boolean hasMoreTokens();
   public String nextToken();
   public String nextToken(String delim);
   public boolean hasMoreElements();
   public Object nextElement();
   public int countTokens();
}
```


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!




```
import java.util.*;
class EventC {
  public static void main (String [] args) {
    int NoOfContestants  = Input.toInt("Enter the number of contestants: ");
    String Name  [] = new String [NoOfContestants];
    int    Score [] = new int    [NoOfContestants];

    for (int c=0; c<NoOfContestants; c++) {
        String S = Input.toText("Enter 'NAME SCORE1 SCORE2 SCORE3' for " +
                                "contestant #" + (c+1) + ":");
        StringTokenizer T = new StringTokenizer(S);
        Name[c] = T.nextToken();
        Score[c] = 0;
        for (int i=0; i<3; i++) {
           String V = T.nextToken();
           Score[c] += Integer.valueOf(V.trim()).intValue();
        }
     }
     // Code for printing table as before!
  }
}
```

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)
```

Slide 5–16

Do and While Loops II

Examples:

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

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

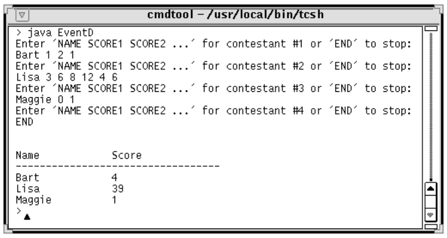
- Sometimes an *indeterminate* loop with a jump out from the middle makes for simpler loop conditions:

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

Slide 5–17

`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!



Slide 5–18

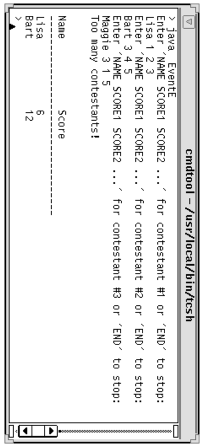
```
class EventD {
   public static int MaxNoOfContestants = 100;
   public static void main (String [] args) {
      String Name  [] = new String [MaxNoOfContestants];
      int    Score [] = new int    [MaxNoOfContestants];
      int Contestants = 0; boolean Done = false;
      do {
         String S = Input.toText("Enter 'NAME SCORE1 SCORE2 ...' for " +
                   "contestant #" + (Contestants+1) + " or 'END' to stop:");
         StringTokenizer T = new StringTokenizer(S);
         String F = T.nextToken(); Done = F.equals("END");
         if (!Done) {
            Name[Contestants] = F; Score[Contestants] = 0;
            while (T.hasMoreTokens()) {
               String V = T.nextToken();
               Score[Contestants] += Integer.valueOf(V.trim()).intValue();
            };
            Contestants++;
         };
      } while (!Done); // Code for printing table as before!
}}
```

Slide 5–19

Event: Break

- 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:

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



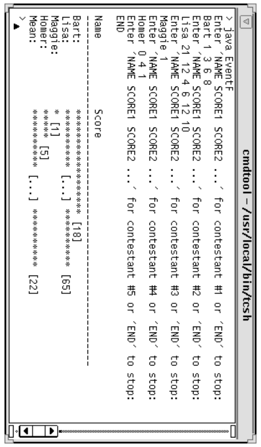
Slide 5–20

```
class EventE {
  public static int MaxNoOfContestants = 2;
  public static void main (String [] args) {
    String Name  [] = new String [MaxNoOfContestants];
    int    Score [] = new int    [MaxNoOfContestants];
    int Contestants = 0;
    while (true) {
        String S = Input.toText(" [...] ");
        StringTokenizer T = new StringTokenizer(S);
        String F = T.nextToken();
        if (F.equals("END")) break;
        if (Contestants == MaxNoOfContestants) {
           System.out.println("Too many contestants!");
           break;
        };
        Name[Contestants] = F;
        // Read the scores using nextToken. As before!
        Contestants++;
     };
     // Printing of table as before!
}
```

Slide 5–21

Event: Histogram I

- This time, print out a $\boxed{\text{horizontal}}$ 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.



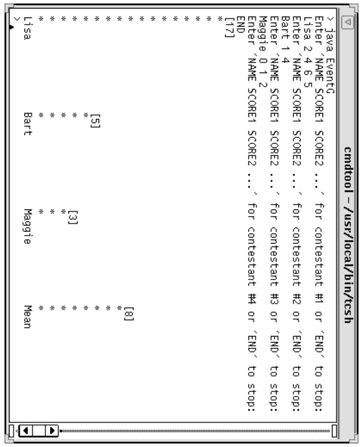
Slide 5–22

```
class EventF {
  public static int MaxScore = 20;
  public static void main (String [] args) {
     // Read in names and scores, as before!
     int Total=0;
     for (int c=0; c<Contestants; c++) Total += Score[c];
     Score[Contestants]= Total/Contestants;
     Name[Contestants] = "Mean"; Contestants++;
     System.out.println("\nName\t\tScore");
     for(int i=1; i<=MaxScore+40; i++) System.out.print("-");
     System.out.println("");
     for (int c=0; c<Contestants; c++){
        System.out.print(Name[c] + ":\t\t");
        if (Score[c] > MaxScore) {
           for(int i=1; i<= MaxScore/2; i++) System.out.print("*");
           System.out.print(" [...] ");
           for(int i=1; i<=MaxScore/2; i++) System.out.print("*");
        } else
           for(int i=1; i<=Score[c]; i++) System.out.print("*");
        System.out.println(" [" + Score[c] + "]");
}}}
```

Slide 5–23

Event: Histogram II

- This time, print out a vertical histogram of the results.
- The program is now 59 lines long!




```
class EventG {
  public static void main (String [] args) {
     // Read in names and scores & compute average, as before!

     int MaxScore=0;
     for (int c=0; c<Contestants; c++)
        MaxScore = Math.max(MaxScore, Score[c]);

     for (int s=MaxScore; s>=0; s--){
        for (int c=0; c<Contestants; c++){
           if      (Score[c] == s) System.out.print("["+Score[c]+"]");
           else if (Score[c] >= s) System.out.print("*");
           System.out.print("\t\t");
        }
        System.out.println();
     }
     for (int c=0; c<Contestants; c++)
        System.out.print(Name[c] + "\t\t");
     System.out.println();
}}
```