Road Map

• while loop
• Looping over Strings
• Sentinels
• Sentinel Controlled Loops
• do/while Loop

• Reading
  – Liang: chapter 4: 4.2, 4.3
review

- When must you use an `if / else if / else` statement instead of using a switch statement?
- When is it preferable to use a switch statement?
- What are three ways to rewrite:
  
  ```
  x = x + 1;
  ```
- What is the difference between pre and post increment operators?
Review continued

• What is the output of this switch statement?

```java
int a = 90;
switch (a)
{
    case 80:
        System.out.println (80);
        break;
    case 90:
        System.out.println (90);
        break;
    case 100:
        System.out.println (100);
        break;
}
```
• Given:
  ```java
  char c = 'a';
  c++;
  ```
  What is the value of variable c?

• What method do you use to extract characters from a String?
  – How would you use that method to get the first character from a String?

• How are characters stored in memory?

• What is the difference between the character ' 0 ' and the integer 0?
Introduction to While Loops
While Loops

- While Loop: Keep repeating an action while some condition remains true.

- Examples:
  - Every Stairmaster Machine contains a while loop (end condition is based on mode used).
    - while the person is still climbing, keep displaying the status, e.g. number of stairs climbed, calories burned, etc.
  - Keep prompting for book orders until the user is done.
while loop (continued)

• For example (in pseudocode)

```plaintext
while (some Boolean expression is true)
{
    do this (again and again...)
}
```
Parts of a While Loop

• Every while loop will always contain three main elements:
  1) **Priming**: initialize your variables.
  2) **Testing**: test against some known condition.
  3) **Updating**: updates part (or all) of the expression that is tested.
public class While1 {
    public static void main (String args[]) {
        int index = 1;  
        while (index <= 10) {
            System.out.println ("Index: " + index);
            index++;
        }
    }
}
1. Priming
   Set index=1

2. Test
   index <= 10

3. Print value of index
   Update index++
Infinite Loop

• Infinite Loop: A loop that never ends.
  – Generally, you want to avoid these!
  – There are special cases, however, when you do want to create infinite loops on purpose.

• Common Exam Questions:
  – Given a piece of code, identify the bug in the code.
  – You may need to identify infinite loops.
Infinite Loop Example #1

public class While2
{
    public static void main (String args[])
    {
        int index = 1;
        while (index <= 10)
        {
            System.out.println ("Index: " + index);
        }
    }
}
Infinite Loop, Example #2

public class While3
{
    public static void main (String args[])
    {
        int index = 1;
        while (index >= 0)
        {
            System.out.println ("Index: "+index);
            index++;
        }
    }
}
While Loops: Examples
While Loop Example

• Specification for the program:
  – Find the first power of 2 larger than 1000.

• For example: 2, 4, 8, 16, 32, etc. are powers of 2.
  – Which is the first power of 2 larger than 1000?
  – Finding the answer to this requires some kind of a while loop.
  – Let’s see how…
• Example:

```java
int product = 2;
while ( product <= 1000 )
    product = 2 * product;
```
public class PowerOfTwoOver1000
{
    public static void main (String args[])
    {
        int product = 2;
        while ( product <= 1000 )
            product = 2 * product;

        System.out.println (product);
    }
}
Counter-Controlled Repetition

• Counter-controlled repetition
  – Loop repeated until counter reaches a certain value
  – Definite repetition: number of repetitions is known
  – Example: A class of ten students took a quiz. The grades (integers in the range 0 to 100) for this quiz are available to you. Determine the class average on the quiz
4.8 Formulating Algorithms: Case Study 1 (Counter-Controlled Repetition)

• Counter
  – Variable that controls number of times set of statements executes

• Average1.java calculates grade averages
  – uses counters to control repetition
Set total to zero
Set grade counter to one

While grade counter is less than or equal to ten
   Input the next grade
   Add the grade into the total
   Add one to the grade counter

Set the class average to the total divided by ten
Print the class average

Fig. 4.6 Pseudocode algorithm that uses counter-controlled repetition to solve the class-average problem.
// Fig. 4.7: Average1.java
// Class-average program with counter-controlled repetition.
import javax.swing.JOptionPane;

public class Average1 {

    public static void main( String args[] )
    {
        int total; // sum of grades input by user
        int gradeCounter; // number of grade to be entered next
        int grade; // grade value
        int average; // average of grades

        String gradeString; // grade typed by user

        // initialization phase
        total = 0; // initialize total
        gradeCounter = 1; // initialize loop counter

        // processing phase
        while ( gradeCounter <= 10 ) { // loop 10 times

            // prompt for input and read grade from user
            gradeString = JOptionPane.showInputDialog(
                "Enter integer grade: ");

            // convert gradeString to int
            grade = Integer.parseInt( gradeString );
        }
    }
}

©2003 Prentice Hall, Inc. All rights reserved.
```java
    total = total + grade;        // add grade to total
    gradeCounter = gradeCounter + 1;  // increment counter

    } // end while

    // termination phase
    average = total / 10;  // integer division

    // display average of exam grades
    JOptionPane.showMessageDialog( null, "Class average is " +
        average, "Class Average", JOptionPane.INFORMATION_MESSAGE );

    System.exit( 0 );  // terminate the program

    } // end main

} // end class Average1
```
Strings
String length()

• `s.length()` tells us how many characters are in the String `s`.

• E.g., the following prints out 5

```
String s = "hello";
System.out.println(s.length());
```

• Recall that `s.charAt(i)` gives us the ith character of `s`

• E.g., the following prints out `e`

```
System.out.println(s.charAt(1));
```
Looping over a String

- We can use `length()` and `charAt()` to loop over the characters in a String

```java
int i = 0;
while( i < s.length() ) {
    char c = s.charAt(i);
    System.out.println("char: " + c);
    System.out.println("ASCII: " + (int)c);
}
```
Sentinels
Sentinel-controlled repetition

• What if we want the user or the input to decide when to end the program?
• Sentinel: a special value that indicates the “end of data entry.”
  – aka signal value, dummy value, or flag value
• For example:
  – -1 means end of data.
  – 0 means end of data.
  – "END" means ends of data
  – Depends on the specific application you are building.
• With a sentinel, we have an indefinite repetition, because the number of repetitions is unknown at the time we write the program (or start the loop).
Using Sentinels

• How are they used?
  – Programmer picks a value that would never be encountered for normal data
  – User enters normal data and then when done, enters the sentinel value
  – The loop will stop when seeing the sentinel value
Using Sentinels cont’d

• For example, if entering age for people, could pick a sentinel of −1
  • No one would expect to be −1 year old.

• Good practice is to remind the user in each iteration of the loop what the sentinel value is
  – For example,
    
    ```java
    System.out.println
    (" Enter age of current resident or –1 to end" );
    ```
import javax.swing.JOptionPane;

public class Sentinel
{
    public static void main(String [] args)
    {
        int currentAge = 0;
        String currentAgeAsString;
        /* priming */
        currentAgeAsString = JOptionPane.showInputDialog ("Enter age of resident: ");
        currentAge = Integer.parseInt (currentAgeAsString);
        /* testing: keep going until input is sentinel value */
        while (currentAge != -1)
        {
            /* do some calculations with age, e.g. AVERAGE */
            /* updating: get the next value from the user */
            currentAgeAsString = JOptionPane.showInputDialog ("Enter age of resident: ");
            currentAge = Integer.parseInt (currentAgeAsString);
        }
        System.exit (0);
    }
}
Good Programming tips

• Pick a sentinel value that you are certain will never be confused with normal data
• Style: Remind user each iteration what the sentinel is
• Y2K-like problem
  – Programmers often used 9999 as a sentinel to end a loop
  – Worry that on September 9, 1999 (sometimes abbreviated 9999) programs would erroneously stop executing before they were supposed to.