Repetition

CSC 1051 – Data Structures and Algorithms I
Dr. Mary-Angela Papalaskari
Department of Computing Sciences
Villanova University

Course website:
www.csc.villanova.edu/~map/1051/

Some slides in this presentation are adapted from the slides accompanying Java Software Solutions by Lewis & Loftus

Control flow: Repetition

- Sequence of statements that are actually executed in a program
- Conditional and Repetition statements: enable us to alter control flow

This slide adapted from Doug Clark’s course http://www.cs.princeton.edu/courses/archive/spring13/cos126/lectures.php

CSC 1051 M.A. Papalaskari, Villanova University
Example

• **Investment problem:** You put $10,000 into a bank account that earns 5% interest per year.

<table>
<thead>
<tr>
<th>year</th>
<th>interest</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>$10,000.00</td>
</tr>
<tr>
<td>1</td>
<td>$500.00</td>
<td>$10,500.00</td>
</tr>
<tr>
<td>2</td>
<td>$525.00</td>
<td>$11,025.00</td>
</tr>
<tr>
<td>3</td>
<td>$551.25</td>
<td>$11,576.25</td>
</tr>
<tr>
<td>4</td>
<td>$578.81</td>
<td>$12,155.06</td>
</tr>
</tbody>
</table>

• … How many years does it take for the account balance to be double the original?

This example is adapted from Cay Horstmann’s *Big Java, Early Objects*, 5th edition

Algorithm:
The while Statement

while ( condition )
statement;

- If the condition is true, the statement is executed.
- Then the condition is evaluated again, and if it is still true, the statement is executed again.
- The statement is executed repeatedly until the condition becomes false.

Example

- Print the even numbers from 2 to 100

Algorithm:

- initialize a counter to 2
- while counter <= 100
  - print counter
  - add 2 to counter

Java? → use while statement
Try this:

• Write a loop to print the powers of 2 less than or equal to 2048, i.e., 2, 4, 8, 16, ..., 1024, 2048.
The **while** statement in action

```java
int count = 1;
while (count <= 3) {
    System.out.println(count);
    count++;
}
```

Output:

1
2
3

CSC 1051 M.A. Papalaskari, Villanova University

The **while** statement "unraveled"

```java
int count = 1;
TEST:(count <= 3) \(\Rightarrow\) true { 
    System.out.println(count);
    count++;
}
TEST:(count <= 3) \(\Rightarrow\) true { 
    System.out.println(count);
    count++;
}
TEST:(count <= 3) \(\Rightarrow\) true { 
    System.out.println(count);
    count++;
}
TEST:(count <= 3) \(\Rightarrow\) false 
    EXIT LOOP
```
What’s wrong with this code?

```java
int count = 1;
while (count <= 10)
    System.out.println (count);
count++;
```

What’s wrong with this code?

```java
int count = 1;
while (count <= 10);
{
    System.out.println (count);
    count++;
}
```
If the condition of a `while` loop is false initially, the statement is never executed.

```java
System.out.println("input a number >5");
int num = scan.nextInt();

// input validation
while (num <= 5) {
    System.out.println("type a number >5");
    num = scan.nextInt();
}
```

- Therefore, the body of a `while` loop will execute zero or more times.

Example: Input validation

```java
System.out.println("Enter lifestyle code");
System.out.println("0=bad; 1=ok; 2=super fit");
int lifestyle = scan.nextInt();

while (lifestyle < 0 || lifestyle > 2) {
    System.out.println("Please try again");
    System.out.println("0=bad; 1=ok; 2=super fit");
    num = scan.nextInt();
}
```
What if we want to do a calculation over and over again?

**Example:** Calculating GPA for many students (how many? when do you stop?)

**Possible scenarios:**
- Keep accepting new inputs (for each student) and calculating and printing corresponding GPA until user quits program (infinite loop).
- Same, but ask each time whether to keep going.
- Same, but quit if the user inputs -1 for the credits (signals end)
- Calculate GPA for exactly 20 students
Example

- Table of squares and cubes:

<table>
<thead>
<tr>
<th>N</th>
<th>N^2</th>
<th>N^3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>64</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>125</td>
</tr>
</tbody>
</table>
Example

- **Table of powers:** Compute the powers of 2 and the powers of 3 and print a table like this:

<table>
<thead>
<tr>
<th>N</th>
<th>2^N</th>
<th>3^N</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>81</td>
</tr>
</tbody>
</table>
Nested loops

**Example: Investment problem repetition**

The repeated action (calculating the number of years it takes for an investment to double) involves repetition.

General pattern for algorithms: A *nested loop*

1. **while** (condition for repeating action)
   - initialize variables
2. **while** (condition for reaching goal)
   - calculations
3. print results

---

Quick Check

How many times will the string "Here" be printed?

```java
count1 = 1;
while (count1 <= 10) {
    count2 = 1;
    while (count2 < 20) {
        System.out.println ("Here");
        count2++;
    }
    count1++;
}
```
Example: Factoring a positive integer

```java
// Finds prime factors of number x.
import java.util.Scanner;
public class PrimeFactors {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.print("Enter number = ");
        int x = scan.nextInt();
        System.out.println("Prime factors of " + x);
        int factor = 2;
        while (factor <= x) {
            while (x % factor == 0) {
                System.out.println(factor);
                x /= factor;
            }
            factor++;
        }
    }
}
```

Example: Graphics (next week)

```java
import java.awt.*;
import javax.swing.JApplet;
public class ManyBoxes extends JApplet {
    public void paint(Graphics page) {
        int red, green, blue;
        red = green = blue = 0; // black
        int size = 500; // largest size
        int count = 10;
        while (count > 0) {
            size = count * 50;
            page.setColor(new Color(red, green, blue));
            page.fillRect(0, 0, size, size);
            count--;
            green += 25;
        }
    }
}
```