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
Topic Thread

• 2.1 Character Strings
• 2.2 Variables, Assignment
• 2.3 Data Types, in particular int, double
• 2.4 Expressions (simple)
• 2.5 Data Conversion
• 2.6 Interactive Programs
• 5.1 Boolean Expressions
• 5.2 The if Statement
• 5.4 The while Statement
Flow of Control

The order of statement execution

• Unless specified otherwise, the order of statement execution through a method is **linear**

• Some programming statements allow us to:
  – decide whether or not to execute a particular statement
  – execute a statement over and over, repetitively

• These decisions are based on *boolean expressions* (or *conditions*) that evaluate to **true** or **false**
Flow of Control

The order of statement execution

• Unless specified otherwise, the order of statement execution through a method is **linear**

• Some programming statements allow us to:
  – decide whether or not to execute a particular statement
  – execute a statement over and over, repetitively

• These decisions are based on **boolean expressions** (or **conditions**) that evaluate to **true** or **false**
The while Statement

- A `while` statement has the following syntax:
  ```
  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
Logic of a while Loop

condition evaluated

true

false

statement

Logic of a while Loop

CSC 1051 M.A. Papalaskari, Villanova University
The while Statement

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

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

```java
int count = 1;
while (count <= 3) {
    System.out.println (count);
    count++; // count <= 3 is true
}
```
The while Statement

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

Output:

1
The while Statement

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

Output:
1
2
3
int count = 1;
while (count <= 3) {
    System.out.println (count);
    count++;
}

Output:
1

CSC 1051 M.A. Papalaskari, Villanova University
int count = 1;
while (count <= 3)
{
    System.out.println (count);
    count++;
}

Output:
1
2
The while Statement

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

Output:
1 2 3
The while Statement

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

**Output:**

```
1
2
```
The while Statement

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

Output:

```
1
2
3
```

Print count
The while Statement

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

**Output:**
1
2
3
The while Statement

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

Output:
1
2
3
The while Statement “unraveled”

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

Output:
1
2
3

CSC 1051 M.A. Papalaskari, Villanova University

int count = 1;
TEST:(count <= 3)⇒ true
{
    System.out.println(count);
    count++;
}
TEST:(count <= 3)⇒ true
{
    System.out.println(count);
    count++;
}
TEST:(count <= 3)⇒ true
{
    System.out.println(count);
    count++;
}
TEST:(count <= 3)⇒ false

EXIT LOOP
GPA problem:
Algorithm for next solution

more = true;

while (more)
{
    input qp
    input credits
    qpa = qp/credits
    print qpa
    print "Enter 1 to continue, 0 to quit" 
    input answer
    more = (1 == answer)
}
int qp;
int credits;
double gpa;
Scanner scan = new Scanner(System.in);

boolean more = true;

while (more)
{
    // get input
    System.out.print ("Enter Quality Points > ");
    qp = scan.nextInt();

    System.out.print ("Enter Credits > ");
    credits = scan.nextInt();

    ... // other logic goes here
    ...

    System.out.print ("Enter 1 to continue, 0 to quit > ");
    more = (1 == scan.nextInt());
}
If the condition of a `while` loop is false initially, the statement is never executed.

```java
int count = 8;
while (count <= 3)
{
    System.out.println (count);
    count++;
}
```

• Therefore, the body of a `while` loop will execute zero or more times.
Example: Input validation

```java
System.out.println("type in a number >5");
int num = scan.nextInt();
while (num <= 5)
{
    System.out.println ("Please try again");
    System.out.println ("type a number >5");
    num = scan.nextInt();
}
```
Example: Input validation

```java
System.out.println("type in a number >5");
int num = scan.nextInt();
while (num <= 5)
{
    System.out.println ("Please try again");
    System.out.println ("type a number >5");
    num = scan.nextInt();
}
```

- In this example, the body of the `while` loop will typically execute zero times
Let’s try this with the Wages.java program

```java
import java.text.NumberFormat;
import java.util.Scanner;

public class Wages {
    // Reads the number of hours worked and calculates wages.
    public static void main (String[] args) {
        final double RATE = 8.25; // regular pay rate
        final int STANDARD = 40; // standard hours in a work week

        Scanner scan = new Scanner (System.in);

        double pay = 0.0;

        continue
    }
}
```
continue

System.out.print ("Enter the number of hours worked: ");
int hours = scan.nextInt();

// Pay overtime at "time and a half"
if (hours > STANDARD)
    pay = STANDARD * RATE + (hours-STDANDARD) * (RATE * 1.5);
else
    pay = hours * RATE;

NumberFormat fmt = NumberFormat.getCurrencyInstance();
System.out.println ("Gross earnings: " + fmt.format(pay));
} }
What if we want to do a calculation over and over again?

- **Example**: Keep calculating wages until user quits program (infinite loop).

- **Example**: Keep calculating wages and ask each time whether to keep going.

- **Example**: Keep calculating wages until user inputs zero for the hours.

- **Example**: Calculate wages for 20 employees.
Homework

• Read Section 5.4, up to Nested loops (pp 230-235)
  
  – **Always** do all self-review exercises when you review material
  
  – *This time you only need to do SR 5.17-SR 5.20 and SR 5.22.*

• Do end of chapter Exercises EX 5.7 – 5.11