Repetition Statements

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
Repetition structures in Java

**while** loop:

```java
int count = 0;
while (count < 5)
{
    count++;
    System.out.println (count);
}
```

- Other repetition structures (Chapter 6 in text)
  - the **do** loop
  - the **for** loop
The **while** and **do** loops are similar.

**while Loop**

1. **condition evaluated**
   - **true** → **statement**
   - **false** → **condition evaluated**

**do Loop**

1. **statement**
   - **true** → **condition evaluated**
   - **false** → **statement**

CSC 1051 M.A. Papalaskari, Villanova University
**for**: a loop with built-in "counter"

- **Initialization**
- **Condition evaluated**
  - **true**: statement
  - **false**: increment

CSC 1051 M.A. Papalaskari, Villanova University
An example of a **do** loop:

```java
int count = 0;
do {
    count++;
    System.out.println (count);
} while (count < 5);
```

- The body of a **do** loop executes at least once
The **do** Statement

- A **do** *statement* has the following syntax:

  ```
  do
  {
   statement-list;
  } while (condition); //end do
  ```

- The *statement-list* is executed once initially, and then the *condition* is evaluated.

- The statement is executed repeatedly until the condition becomes **false**.
Try this:

• Write a do loop to print the even numbers from 2 to 100.
Try this:

• Write a do loop to keep repeating a computation while a user has more data.
import java.util.Scanner;

public class GradeReportLoopy
{
  //-------------------------------------------------------------------------------
  //  Reads grades from the user and prints comments accordingly.
  //-------------------------------------------------------------------------------
  public static void main (String[] args)
  {
    int grade;
    String ans;

    Scanner scan = new Scanner (System.in);
    do
    {
      System.out.print ("Enter a numeric grade (0 to 100): ");
      grade = scan.nextInt();
      printReport(grade);
      System.out.print ("Would you like to do another grade report? ");
      ans = scan.next();
    } while (ans.charAt(0) == 'y' || ans.charAt(0) == 'Y') ;

    continue
  }
}
public static void printReport (int grade) {
    int category = grade / 10;

    System.out.print ("That grade is ");

    switch (category) {
    case 10:
        System.out.println ("a perfect score. Well done.");
        break;
    case 9:
        System.out.println ("great!");
    case 8:
        System.out.println ("well above average. Excellent.");
        break;
    case 7:
        System.out.println ("above average. Nice job.");
        break;
    case 6
        System.out.println ("below average.");
        break;
    default:
        System.out.println ("not passing.");
    }
}

import java.util.Scanner;

public class GradeReportLoopy {
    public static void main(String[] args) {
        int grade;
        String ans;

        Scanner scan = new Scanner(System.in);
        do {
            System.out.print("Enter a numeric grade (0 to 100): ");
            grade = scan.nextInt();
            printReport(grade);
            System.out.print("Would you like to do another grade report? ");
            ans = scan.next();
        } while (ans.charAt(0) == 'y' || ans.charAt(0) == 'Y');
    }
}

Output

Enter a numeric grade (0 to 100): 84
That grade is well above average. Excellent.
Would you like to do another grade report? yes
Enter a numeric grade (0 to 100): 99
That grade is great!
well above average. Excellent.
Would you like to do another grade report? Y
Enter a numeric grade (0 to 100): 100
That grade is a perfect score. Well done.
Would you like to do another grade report? no
Try this:

This code inputs a single grade:

```java
System.out.print("Enter a numeric grade (0 to 100): ");
grade = scan.nextInt();
```

Use a do loop to do **input verification** – i.e., ensuring that the grade typed in is in the range 1-100 (ask them to keep trying again and again until they input a number in the range)
The \texttt{for} Statement

• A \texttt{for} loop is functionally equivalent to the following \texttt{while} loop structure:

\begin{verbatim}
initialization;
while ( condition )
{
    statement;
    increment;
}
\end{verbatim}
The **for** Statement - Example

• A **while** loop:

```java
int count=1;
while (count <= 5)
{
    System.out.println (count);
    count++;
}
```

• Equivalent **for** loop:

```java
for (int count=1; count <= 5; count++)
    System.out.println (count);
```
The for Statement

• A *for statement* has the following syntax:

```
for ( initialization ; condition ; increment )
  statement;
```

- The *initialization* is executed once before the loop begins.
- The *statement* is executed until the *condition* becomes false.
- The *increment* portion is executed at the end of each iteration.
The for Statement

• A *for statement* has the following syntax:

```java
for (int count=1; count <= 5; count++)
    System.out.println (count);
```

The *initialization* is executed once before the loop begins.

The *statement* is executed until the *condition* becomes false.

The *increment* portion is executed at the end of each iteration.
Try this:

- Write a for loop to print the even numbers from 2 to 100.
Character data reminder

Character data is stored as numeric codes. In Java, a char is stored as a 16-bit number – same as a short integer. We can convert back and forth between integers and characters using casts. Note that going between char and short is a narrowing conversion in both directions.

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>char</td>
</tr>
<tr>
<td>short</td>
<td>byte or char</td>
</tr>
<tr>
<td>char</td>
<td>byte or short</td>
</tr>
<tr>
<td>int</td>
<td>byte, short, or char</td>
</tr>
<tr>
<td>long</td>
<td>byte, short, char, or int</td>
</tr>
<tr>
<td>float</td>
<td>byte, short, char, int, or long</td>
</tr>
<tr>
<td>double</td>
<td>byte, short, char, int, long, or float</td>
</tr>
</tbody>
</table>

(Figure 2.6 page 84 in textbook – Java narrowing conversions)
Try this:

- Write a loop to print out the characters corresponding to the ASCII codes 0-255
Try this:

- Write a loop to print out the numbers (ASCII codes) corresponding to the characters ‘A’ through ‘Z’
Try this:

Write a for loop to print n stars, where n is an integer parameter
The for Statement

• The increment section can perform any calculation:

```java
for (int num=100; num > 0; num -= 5)
    System.out.println (num);
```

• A `for` loop is well suited for executing statements a specific number of times that can be calculated or determined in advance

• See `<Stars.java>`
public class Stars {
    public static void main (String[] args) {
        final int MAX_ROWS = 10;

        for (int row = 1; row <= MAX_ROWS; row++) {
            for (int star = 1; star <= row; star++)
                System.out.print("*");

            System.out.println();
        }
    }
}
public class Stars {
    // Prints a triangle shape using asterisk (star) characters.
    public static void main(String[] args) {
        final int MAX_ROWS = 10;
        for (int row = 1; row <= MAX_ROWS; row++) {
            for (int star = 1; star <= row; star++)
                System.out.print("*");
            System.out.println();
        }
    }
}
The for Statement

- Each expression in the header of a `for` loop is optional
- If the initialization is left out, no initialization is performed
- If the condition is left out, it is always considered to be true, and therefore creates an infinite loop
- If the increment is left out, no increment operation is performed
Which loop to use???

• We need to write a program that finds the largest of some numbers input through the keyboard (or some other such calculation involving repeatedly getting input).

• Which type of loop should we use if…
  a) We ask the user whether to continue, each time through the loop
  b) We know in advance how many numbers there will be?
  c) We don’t know in advance how many, but don’t want to ask each time through the loop either.
Homework

- Review Sections 6.3, 6.4, and 6.5
  - **Always** do all self-review exercises when you review
- Exercises EX6.1 - 6.17 (important to get lots of practice with writing loops!)
- Look at some additional examples of programs from the text:
  - [ReverseNumber.java](#)
  - [Multiples.java](#)