Selection and Repetition
Revisited

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
Selection structures in Java

• Conditional statement:

  ```java
  if (n > 0)
      System.out.print("positive");
  else
      System.out.print("negative");
  ```

• Other selection structures (Chapter 6 in text)
  – the `conditional` operator
  – the `switch` statement
The Conditional Operator Syntax

\[ \text{condition } ? \text{ expression1 } : \text{ expression2} \]

- If the \textit{condition} is true, \textit{expression1} is evaluated; if it is false, \textit{expression2} is evaluated
- The value of the entire conditional operator is the value of the selected expression
- Example: Rewrite this

```java
if (n > 0)
    System.out.print("positive");
else
    System.out.print("negative");
```
The conditional operator is **not a statement**

```java
(n > 0) ? System.out.print("positive") : System.out.print("negative");
```
More examples:

```java
int bit = (ans.equalsIgnoreCase("Yes") ? 1 : 0);
String status = (age < 18 ? "child" : "adult");
```

- The conditional operator requires **three** operands so it is sometimes called the **ternary** operator
Try this:

*Rewrite this statement so that "Dime" is printed if num equals 1.*

```java
System.out.println ("Your change is " + num + "Dimes");
```

use conditional operator:
The **switch** Statement: multi-way branches

Recall: Logic of an if-else statement

- Condition evaluated
  - True: execute `statement1`
  - False: execute `statement2`
The *switch* Statement: multi-way branches

**switch statement logic**

Note: this is a simplified flowchart of the logic of the switch statement

CSC 1051 M.A. Papalaskari, Villanova University
Random rand = new Random();
int number = rand.nextInt(3) + 1;

String word = "";
switch (number)
{
    case 1:
        word = "one";
        break;
    case 2:
        word = "two";
        break;
    case 3:
        word = "three";
        break;
}
System.out.println(“You are my number “ + word);
The switch Statement in general

```java
switch (expression) {
    case value1:
        statement-list1
    case value2:
        statement-list2
    case value3:
        statement-list3
    .
    .
    .
    default:
        statement-list-n
}
```

- Integer, char, or enumerated types (Java 7 also allows Strings)
- NO floating point values
- NO ranges of values (eg: 0<x<10)

If `expression` matches `value2`, control jumps to here.

If none of the values match the `expression`, control jumps to here.

CSC 1051 M.A. Papalaskari, Villanova University
So... the logic of the switch is more like this:

Note: this is still a simplified flowchart of the logic of the switch statement
import java.util.Scanner;

public class GradeReport{
    //--
    // Reads a grade from the user and prints comments accordingly.
    //--
    public static void main (String[] args){
        int grade, category;

        Scanner scan = new Scanner (System.in);

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

        category = grade / 10;

        System.out.print ("That grade is ");
        continue
```java
switch (category) {
    case 10:
        System.out.println("a perfect score. Well done.");
        break;
    case 9:
        System.out.println("well above average. Excellent.");
        break;
    case 8:
        System.out.println("above average. Nice job.");
        break;
    case 7:
        System.out.println("average.");
        break;
    case 6:
        System.out.println("below average. You should see the");
        System.out.println("instructor to clarify the material " + "presented in class.");
        break;
    default:
        System.out.println("not passing.");
}
```

Sample Run

Enter a numeric grade (0 to 100): 91
That grade is well above average. Excellent.

Hands on: try removing one of the break statements
import java.util.Scanner;

public class SwitchExample {
    // Example of using a switch statement. Counts number of digits, zeros, whitespace, and others in a line of input.
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int digits = 0, zeros = 0, whitespace = 0, other = 0;
        System.out.print("Input line>");
        String message = scan.nextLine();
        int count = 0;
        continue
        Sample Run
        Input line>10, 9, 8, 7, 6, 5, 4, 3, 2, 1, Lift off!
        11 Digits, of which 1 is a zero
        11 whitespace
        18 others
```java
while (count < message.length()) {
    switch (message.charAt(count)) {
        case '0':    zeros++;    
        case '1':    
        case '2':    
        case '3':    
        case '4':    
        case '5':    
        case '6':    
        case '7':    
        case '8':    
        case '9':     digits++;    break;
        case ' ':     whitespace++;    break;
        default:     other++;
    } // end switch
    count++;
} // end while
System.out.print(digits + " Digit" + (digits==1 ? "" : "s"));
System.out.print(" , of which " + zeros);
System.out.println((zeros==1 ? " is a zero " : " are zeros ") );

System.out.println(whitespace + " whitespace");
System.out.println(other + " other" + (other==1 ? "" : "s"));
}
```
Loops revisited: do & for loops

Repetition structures in Java, so far:

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

• Other repetition structures (Chapter 6 in text)
  – the do loop
  – the for loop
The **do** Statement in Java

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

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

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

- The **statement-list** is executed **repeatedly** until the condition becomes **false**.
The **while** and **do** loops are similar.

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

```java
int count = 0;
do {
    System.out.println (count);
    count++;
} while (count < 5);
```
Similar – but not the same:

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

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

- The body of a `do` loop executes **at least once**
Try this:

• Write a do loop to print the even numbers from 2 to 100.
For some things the **do** loop is more appropriate:

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

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

```java
do
{
    System.out.println("type a number >5");
    num = scan.nextInt();
} while (num <= 5);
```
For some things the **do** loop is more appropriate:

```java
int answer = 1;
while (answer == 1)
{
    System.out.print("Enter Credits ");
    credits = scan.nextInt();
    System.out.print("Enter QP ");
    qp = scan.nextInt();

    gpa = (double) qp / credits;
    System.out.println("GPA = " + gpa);
    System.out.print("Again? 1=yes,0=no ");
    answer = scan.nextInt();
}
System.out.println("Thank you. Goodbye.");
```

**repeating a computation**

**For some things the do loop is more appropriate:**

```java
do {
    System.out.print("Enter Credits ");
    credits = scan.nextInt();
    System.out.print("Enter QP ");
    qp = scan.nextInt();

    gpa = (double) qp / credits;
    System.out.println("GPA = " + gpa);
    System.out.print("Again? 1=yes,0=no ");
    answer = scan.nextInt();
}
while (answer == 1);
System.out.println("Thank you. Goodbye.");
```
import java.util.Scanner;
public class ReverseNumber {
    public static void main(String[] args) {
        int number, lastDigit, reverse = 0;
        Scanner scan = new Scanner(System.in);
        System.out.print("Enter a positive integer: ");
        number = scan.nextInt();
        do {
            lastDigit = number % 10;
            reverse = (reverse * 10) + lastDigit;
            number = number / 10;
        } while (number > 0);
        System.out.println("That number reversed is "+ reverse);
    }
}
for: a loop with built in “counter”

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

- **Initialization**: `int count = 0;`
- **Condition**: `while (count < 5)`
- **Statement**: `System.out.println (count);`
- **Increment**: `count++;`

**Example**

```java
int count = 0;

while (count < 5)
    {
        System.out.println (count);
        count++;
    }
```
The **for** Statement

- A **for** loop is functionally equivalent to the following **while** loop structure:

```
initialization;
while ( condition )
{
    statement;
    increment;
}
```

```
for ( initialization ; condition ; increment )
    statement;
```
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 *condition* is tested after the *initialization* and again before each iteration.
  - The *increment* is executed at the end of each iteration.
  - The *statement* is executed while the *condition* is true.

  The *for* statement is executed while the *condition* is true.
The **for** Statement - Example

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

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

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

The **condition** is tested after the **initialization** and again before each iteration.

The **increment** is executed at the end of each iteration.

The **statement** is executed while The **condition** is true.
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.
Try this:

• Write a for loop to print the even numbers from 2 to 100.
Example: ReverseNumberAsString.java

import java.util.Scanner;

public class ReverseNumberAsString {
    public static void main (String[] args) {
        int number;
        String reverse = "";
        Scanner scan = new Scanner (System.in);

        System.out.print ("Enter a positive integer: ");
        number = scan.nextInt();
        String original = Integer.toString(number);

        for (int i=0; i<original.length(); i++)
            reverse = original.charAt(i) + reverse;

        System.out.println ("That number reversed is " + reverse);
    }
}

Sample Run

Enter a positive integer: 2896
That number reversed is 6982
Example: Stars.java

```java
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();
        }
    }
}
```

Output
```
* 
** 
*** 
**** 
***** 
****** 
******* 
******** 
********* 
**********
```

Exercise: can you make it print the row number in (1, 2, 3...) at the beginning of each line?