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

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

WRONG!
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

Hands on: try this in the **Person** class!
Try this:

Using if/else:

```java
if (happiness)
    page.drawArc (x+15, y+30, 20, 10, 180, 180);
else
    page.drawArc (x+15, y+30, 20, 10, 0, 180);
```

– Rewrite using conditional 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:

```java
if (num == 1) {
    System.out.println("Your change is a Dime");
} else {
    System.out.println("Your change is " + num + "Dimes");
}
```
Quick Check

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

**Rewrite this statement so that "Dime" is printed if num equals 1.**
The `switch` Statement: multi-way branches

Recall: Logic of an if-else statement

![Diagram showing the logic of a switch statement](image-url)
The **switch** Statement: multi-way branches

**switch** statement logic

Note: this is a simplified flowchart of the logic of the switch statement
The switch Statement - example

```java
public String toString()
{
    String result = "";
    switch (faceValue)
    {
        case 1:
            result = "one";
            break;
        case 2:
            result = "two";
            break;
        case 3:
            result = "three";
            break;
    }
    return result;
}
```

Hands on: try this in the `Die` class!
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 (e.g., 0 < x < 10)

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

*If none of the values match the expression, control jumps to here*
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
{

    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
    }
}
```
continue

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;

    }
}

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
while (count < message.length()) {
    switch (message.charAt(count)) {
        case '0': zeros++; break;
        case '1':
        case '2':
        case '3':
        case '4':
        case '5':
        case '6':
        case '7':
        case '8':
        case '9': digits++; break;
        case ' ': 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"));
}
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:

- The body of a do loop executes at least once.

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

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

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

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

```
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 QP ");
    qp = scan.nextInt();

    System.out.print ("Enter Credits ");
    credits = 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**
Another example: ReverseNumber.java

//****************************************************************************
// ReverseNumber.java Author: Lewis/Loftus
// Demonstrates the use of a do loop.
//****************************************************************************
import java.util.Scanner;
public class ReverseNumber
{
    //----------------------------------------------------------------------------
    // Reverses the digits of an integer mathematically.
    //----------------------------------------------------------------------------
    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);
    }
}

Sample Run
Enter a positive integer: 2896
That number reversed is 6982
import javax.swing.JOptionPane;

public class EvenOdd
{
    public static void main (String[] args)
    {
        String numStr, result;
        int num, again;
        do
        {
            numStr = JOptionPane.showInputDialog("Enter an integer: ");
            num = Integer.parseInt(numStr);

            result = "That number is " + ((num%2 == 0) ? "even" : "odd");

            JOptionPane.showMessageDialog(null, result);
            again = JOptionPane.showConfirmDialog(null, "Do Another?");
        }
        while (again == JOptionPane.YES_OPTION);
    }
}
import javax.swing.JOptionPane;

public class EvenOdd {
    public static void main (String[] args) {
        String numStr, result;
        int num, again;
        do {
            numStr = JOptionPane.showInputDialog("Enter an integer: ");
            num = Integer.parseInt(numStr);
            result = "That number is " + ((num%2 == 0) ? "even" : "odd");
            JOptionPane.showMessageDialog(null, result);
            again = JOptionPane.showConfirmDialog(null, "Do Another?");
        } while (again == JOptionPane.YES_OPTION);
    }
}
**for**: a loop with built in “counter”

```
initialization

condition evaluated

true

statement

false

increment
```
**for**: a loop with built in “counter”

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

CSC 1051 M.A. Papalaskari, Villanova University
for: a loop with built in “counter”

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

```
int count = 0;
while (count < 5)
{
    { System.out.println (count);
        count++; } // example
```
The **for** Statement

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

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

```java
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 - 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`

```java
//********************************************************************
// ReverseNumberAsString.java  Author: MAP
// Demonstrates the use of a for loop.
//********************************************************************
import java.util.Scanner;

public class ReverseNumberAsString
{
    //-----------------------------------------------------------------
    // Reverses the digits of an integer viewed as a String.
    //-----------------------------------------------------------------
    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:** `ReverseNumberAsString.java`
Example: **Stars.java**

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

**Output**

```
*  
** 
***  
**** 
*****  
******  
*******  
********  
*********  
**********  
***********  
```

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