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/

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
condition ? expression1 : expression2
```
• If the condition is true, expression1 is evaluated; if it is false, 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**

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:

```java
int bit = (ans.equalsIgnoreCase("Yes") ? 1: 0);
```

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

Dr. Papalaskari
The switch Statement: multi-way branches

switch statement logic

---

The switch Statement - example

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

---

The switch Statement in general

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

---

So... the logic of the switch is more like this:

---

Note: this is a simplified flowchart of the logic of the switch statement
### GradeReport.java

```java
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
        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("above 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

### SwitchExample.java

```java
import java.util.Scanner;
public class SwitchExample {
    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
        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++;
            }
            count++;
        }
        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"));
    }
}
```

### Another example

```java
import java.util.Scanner;
public class SwitchExample {
    public static void main (String[] args) {
        Scanner scan = new Scanner(System.in);
        int count = 0;
        continue
        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++;
            }
            count++;
        }
        System.out.print(digits + " Digit" + (digits==1 ? "" : "s"));
        System.out.print(, of which " + zeros); System.out.println(" + is a zero ");
        System.out.println(" + are zeros ");
        System.out.println(whitespace + " whitespace";
        System.out.println(other + " other" + (other==1 ? "" : "s"));
    }
}
```

### 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
```
Selection and Repetition, revisited

Repetition structures in Java, so far:

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

```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:

```java
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);
```

input validation

Another example: `ReverseNumber.java`

```java
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;
            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();
        } while (1 == scan.nextInt());
        System.out.println("Thank you. Goodbye.");
    }
}
```

Sample Run

Enter a positive integer: 2896
That number reversed is 6982
Selection and Repetition, revisited

Another example: EvenOdd.java

```java
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);
    }
}
```

Another example: EvenOdd.java

The JOptionPane class provides methods for creating dialog boxes.

for: a loop with built in “counter”

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

CSC 1051 Villanova University
Selection and Repetition, revisited

The `for` Statement

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

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

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

The `for` Statement - Example

- A `for` statement has the following syntax:

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

The `for` Statement

- A `for` statement has the following syntax:

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

```java
for (int i=0; i<100; i++)
    System.out.println (i);
```

Example: `ReverseNumberAsString.java`

```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 {
    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 ("\n");
        }
    }
}
```

Output

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