Selection and Repetition, revisited

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

```
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 →

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

The conditional operator is not a statement

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

Wrong!
Selection and Repetition, revisited

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:

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

The switch Statement: multi-way branches

```java
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)

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

```
expression evaluated
    1
    2
    3
    statements1
    statements2
    statements3
```

In GradeReport.java:

```java
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 ("average.");
                break;
            case 6:
                System.out.println ("below average. You should see the 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
The switch Statement – another example

```java
Random rand = new Random();
int number = rand.nextInt(100);
String word = "";
switch (number) {
case 0:
    word = "none"; break;
case 1:
    word = "one"; break;
case 2: case 3:
    word = "a few"; break;
case 4: case 5: case 6: case 7:
    word = "several"; break;
default:
    word = "many";
}
System.out.println("At closing time there are " + word + " of the customers left in the queue");
```

Another example

```java
import java.util.Scanner;
public class SwitchExample {
    // Example of 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;
        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++;
            }
        }
        System.out.print(digits + " Digit" + (digits == 1 ? "" : "s"));
        System.out.print(" of which "+zeros + " is a zero " + (zeros == 1 ? " are zeros " : " 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:
  ```java
  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:

**while Loop**

- 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:

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

**input validation**

```
Sample Run
Enter a positive integer: 2896
That number reversed is 6982
```

**Another example: ReverseNumber.java**

// ReverseNumber.java  Author: Lewis
// 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);
    }
}
```

**repeating a computation**

```
System.out.println("Enter Credits ");
credits = scan.nextInt();
System.out.print("Enter QP ");
qp = scan.nextInt();
gpa = (double) qp /credits;
System.out.print("Again? 1=yes,0=no ");
answer = scan.nextInt());
if (answer == 1) { do {
    System.out.print("Enter Credits ");
credits = scan.nextInt();
    System.out.print("Enter QP ");
    qp = scan.nextInt();
    gpa = (double) qp /credits;
    System.out.print("Again? 1=yes,0=no ");
    answer = scan.nextInt());
} while (answer == 1); System.out.println("Thank you. Goodbye.");
```

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

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

**Example**

```
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:

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

The **for** Statement - Example

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

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

  The statement is executed while the condition is true.

  Example

  ```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.

  Example

  ```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
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
```