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:
  \[
  \text{if} \ (n > 0) \\
  \quad \text{System.out.print}("positive"); \\
  \text{else} \\
  \quad \text{System.out.print}("negative");
  \]

- Other selection structures (Chapter 6 in text)
  - the \textit{conditional} operator
  - the \textit{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");
```
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

**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:

Quick Check

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

```java
System.out.println ("Your change is " + num + "Dimes");
```
The switch Statement: multi-way branches

Recall: Logic of an if-else 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;
```

Hands on: try this in the Die class!

```java
    }
    return result;
}
```

The switch Statement in general

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

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

If expression matches value2, control jumps to here

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

Hands on: try this in the Die class!
Selection and Repetition, revisited

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

```
expression evaluated

1 2 3

statements1

statements2

statements3
```

Note: this is still 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("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.");
                break;
        }
    }
}
```

---

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

---

Another example

```
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
        switch (count) {
            case 0:
                System.out.println("11 Digits, of which 1 is a zero");
                System.out.println("11 whitespace");
                System.out.println("18 others");
                break;
            default:
                System.out.println("Sample Run");
                System.out.println("Input line>");
                System.out.println("11 Digits, of which 1 is a zero");
                System.out.println("11 whitespace");
                System.out.println("18 others");
                break;
        }
    }
}
```

---

Dr. Papalaskari
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++;
}
// end switch
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"));
}
// end while

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

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 while and do loops are similar.
Similar – but not the same:

- **while Loop**
  - Condition evaluated
  - true
  - false
  - statement
  - 
  ```java
  int count = 0;
  while (count < 5) {
    System.out.println(count);
    count++;
  }
  ```

- **do Loop**
  - Condition evaluated
  - true
  - statement
  - false
  - 
  ```java
  int count = 0;
  do {
    System.out.println(count);
    count++;
  } while (count < 5);
  ```

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

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();
}
```
Selection and Repetition, revisited

**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;
            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

---

**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:**

```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;
            number = number / 10;
        } while (number > 0);
        System.out.println("That number reversed is "+ reverse);
    }
}
```

**Another example:**

```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);
    }
}
```

---

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

```
Initialization
condition evaluated
statement
false
true
Increment
```

---

The `JOptionPane` class provides methods for creating dialog boxes.

---

Dr. Papalaskari
Selection and Repetition, revisited

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

```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:

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

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

for (initialization; condition; increment) 
statement;

the statement is executed while the condition is true.

The increment is executed at the end of each iteration.
```
Selection and Repetition, revisited

The **for** Statement - Example

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

```java
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.

**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
//********************************************************************
// Stars.java  Author: Lewis/Loftus
// Demonstrates the use of nested for loops.
//********************************************************************
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 (*");
        }
    }
}
```

**Output**

```
*  
** */  
*** ****  
**** *****  
****** ******  
******* *******  
```

Example: *Stars.java*

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