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/

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
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**

```java
(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:

```
System.out.println("Your change is " + (num == 1 ? "Dime" : "Dimes"));
```

The **switch** Statement: multi-way branches

Recall: Logic of an if-else statement

```
switch expression {
    case 1: statement1;
    case 2: statement2;
    case 3: statement3;
    default: statementD;
}
```

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

```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
}
```

The logic of the switch is more like this:

```
expression evaluated

1 2 3

statements1

statements2

statements3
```

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

```
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
    }

    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
        }
    }
```
Selection and Repetition, revisited

Hands on: try removing one of the break statements

Another example

SwitchExample.java

Repetition structures in Java, so far:

- 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:

```
  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.

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

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

---

Similar – but not the same:

- **while** Loop
- **do** Loop

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

```
  do
  {
    statement;
  } while (condition);
```

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

**Input validation**

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

**Repeating a computation**

```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("Again? 1=yes,0=no ");
    answer = scan.nextInt();
}
System.out.println("Thank you. Goodbye.");
```

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

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

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

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

The **for** Statement - Example

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

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

- The **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
import java.util.Scanner;
public class ReverseNumberAsString {
    public static void main (String[] args) {
        int number;
        String reverse = "")
            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

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