JC2002 Java Programming

Lecture 5: Conditional structures and loops

Conditional statements

- In many situations, the program needs to make comparisons to decide what to do next
- In Java, conditional actions are usually implemented with *if … then* structure

Condition: Boolean expression (e.g. comparison) or boolean variable

```
if (condition) {
    do this
}
else {
    do that
}
```

If condition is true, do this

If condition is false, do that

Curly brackets

- Note that Java allows to omit the curly brackets when writing *if* statements containing only a single statement
- However, it is recommended always to use them to avoid bugs that are difficult to notice

✓

```
if (x>y) {
    System.out.println("x>y");
}
```

✓

```
if (x>y)
    System.out.println("x>y");
```

✗

```
if (x>y);
    System.out.println("x>y");
```

Comparisons example with *if* structure

```
// Example of comparison with if
import java.util.Scanner; // needed for input
public class ComparisonIf {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter x:  "); int x = input.nextInt();
        System.out.print("Enter y:  "); int y = input.nextInt();
        if( x == y ) {
            System.out.printf("%d == %d\n", x, y);
        }
        if( x < y ) {
            System.out.printf("%d < %d\n", x, y);
        }
        if( x > y ) {
            System.out.printf("%d > %d\n", x, y);
        }
    } // end section main
} // end class ComparisonIf
```

```
Enter x:  5
Enter y:  5
5 == 5
```

```
Enter x:  0
Enter y:  1
0 < 1
```

```
Enter x:  55
Enter y:  10
55 > 10
```

Boolean operators

- You can combine conditions using Boolean operators
  ! (NOT), && (AND), and || (OR)

```
if (x > a && y > a) {
    System.out.println("x > a and y > a!");
}
if (x > a || y > a) {
    System.out.println("x > a or y > a!");
}
if (!(x > y)) {
    System.out.println("x <= y!");
}
```

1495 UNIVERSITY OF ABERDEEN

Java *if* ... *else* statements

- Most Java programmers prefer to write the preceding nested *if…else* statement as:

```
if (studentGrade >= 90) {
   System.out.println("A");
}
else if (studentGrade >= 80) {
   System.out.println("B");
}
else if (studentGrade >= 70) {
   System.out.println("C");
}
else if (studentGrade >= 60) {
   System.out.println("D");
}
else {
   System.out.println("F");
}
```

Note: By convention, variable-name identifiers in Java use the *camel-case* naming convention with a lowercase first letter (e.g. **`firstNumber`**, **`studentGrade`**).

1495 UNIVERSITY OF ABERDEEN

Java *switch* … *case* statements

- Sometimes it is reasonable to use *switch* … *case* structure instead of multiple comparisons:

```
switch (control) {
    case 0:
        do something
        break;
    case 1:
        do something else
        break;
   default:
        error handling etc.
        break;
}
```

Variable `control` will be tested for different conditions

If the variable has value 0, this will be done

If the variable has value 1, this will be done

This is done by default if none of the defined cases match

Exits the switch

Default action

Comparisons example with *if* structure

```java
// Example of conditional statements with case
import java.util.Scanner; // needed for input
public class TestCase {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Choose 1 or 2: ");  int value = input.nextInt();
        switch(value) {
            case 1:
                System.out.println("You chose 1!");
                break;
            case 2:
                System.out.println("You chose 2!");
                break;
            default:
                System.out.println("You did not choose 1 or 2!");
                break;
        }
    } // end section main
} // end class TestCase
```

```
Choose 1 or 2: 1
You chose 1!
```

```
Choose 1 or 2: 2
You chose 2!
```

```
Choose 1 or 2: 3
You did not choose 1 or 2!
```

1495 UNIVERSITY OF ABERDEEN

Precedence and associativity of operators

Operators	Associativity	Type
`*  /  %`	left to right	multiplicative
`+  -`	left to right	additive
`<  <=  >`	left to right	relational
`==  !=`	left to right	equality
`=`	right to left	assignment

1495 UNIVERSITY OF ABERDEEN

Conditional operator (?:)

- *Conditional operator* (`?:`) is a shorthand `if…else`
    - *Ternary operator* (takes *three* operands)
- Operands and `?:` form a *conditional expression*
    - Operand to the left of the `?` is a *`boolean` expression*—evaluates to a `boolean` value (`true` or `false`)
    - Second operand (between the `?` and `:`) is the value if the `boolean` expression is `true`
    - Third operand (to the right of the `:`) is the value if the `boolean` expression evaluates to `false`

```
System.out.println(studentGrade >= 60 ? "Passed" : "Failed");
```

<-- Boolean expression -->   <-- if true -->   <-- if false -->

Example of conditional operator

```java
// Example of conditional operator
import java.util.Scanner; // needed for input
public class ClassCond {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Write your age: ");
        int age = input.nextInt();
        String str = age < 18 ? "minor" : "adult";
        System.out.printf("You are %s!\n", str);
    } // end section main
} // end class ClassCond
```

```
Write your age: 10
You are minor!
```

```
Write your age: 50
You are adult!
```

- Conditional operator allows you to write compact code, but beware: it can make code difficult to read and prone to bugs!

Java iteration statement *while*

- In some situations, the program needs to repeat an action many times if the condition remains true
- In Java, *while* iteration statement can be used for this

Condition: Boolean expression (e.g. comparison) or boolean variable

```
while (condition) {
    do this
}
continue here
```

If condition is true, do this, then test condition again, if it is still true, repeat doing this

If condition is false, just continue here

Example of using *while* iteration statement

- Find the first power of 3 larger than 100:

```
product = 3;
while (product <= 100)
     product = 3 * product;
```

- Each iteration multiplies `product` by 3, so `product` takes on the values 9, 27, 81 and 243 successively
- When `product` becomes 243, `product <= 100` becomes false
- Iteration terminates, the final value of `product` is 243
- Program execution continues with the next statement after the `while` statement

1495 UNIVERSITY OF ABERDEEN

Example of *while* loop

```
// Example of while loop
import java.util.Scanner; // needed for input
public class ClassAverage {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int total = 0; // initialize sum of grades
        int gradeCounter = 1; // initialize number of grades

        while (gradeCounter <= 10) { // loop ten times
            System.out.print ("Enter grade: ");
            int grade = input.nextInt();
            total = total + grade;
            gradeCounter = gradeCounter + 1;
        }
    } // end section main
} // end class ClassAverage
```

UNIVERSITY OF ABERDEEN

Java *do … while* iteration statement

- The iteration statement do…`while` is similar to `while` statement
- In the `while` statement, loop-continuation condition tested at the *beginning* of the loop, ***before*** executing the loop's body; if the condition is *false*, the body *never* executes
- The do…`while` statement tests the loop-continuation condition ***after*** executing the loop's body; therefore, *the body always executes at least once*
- When a do…`while` loop terminates, execution continues with the next statement in sequence

Example of *do…while* loop

```java
// Example of do..while loop
public class DoWhileTest {
    public static void main(String[] args) {
        int counter = 1; // initialize counter

        do {
            System.out.printf("%d ", counter);
            ++counter;
        } while (counter <= 10);

        System.out.println();
    } // end section main
} // end class DoWhileTest
```

```
1 2 3 4 5 6 7 8 9 10
```

Java *for* loop

- *For* loops are common in many programming languages
- In Java, *for* loops have the following header components:

Initialization expression

Test expression

Update expression

```
for( int counter = 1; counter <= 10; counter++ )
```

Initializes control variable with value 1 when the loop is started

Loop is repeated if the test expression is true

Update expression updates the control variable in each iteration

Example of *for* loop

```java
// Example of for loop
public class ForTest {
    public static void main(String[] args) {

        for (int counter = 1; counter <= 10; counter++) {
            System.out.printf("%d ", counter);
        }

        System.out.println();
    } // end section main
} // end class ForTest
```

```
1 2 3 4 5 6 7 8 9 10
```

1495 UNIVERSITY OF ABERDEEN

Questions, comments?