Programming with Java
CE 274

Course Lecturer: Vincent M. Nofong, Ph.D.

Computer Science and Engineering Department
University of Mines and Technology, Tarkwa - Ghana

February 21, 2019

The if Statements

- An if statement is a construct that enables a program to specify alternative paths of execution.
- Java has several types of selection statements:
  - one-way if statements
  - two-way if-else statements,
  - nested if statements
  - multi-way if-else statements
  - switch statements, and
  - conditional expressions.

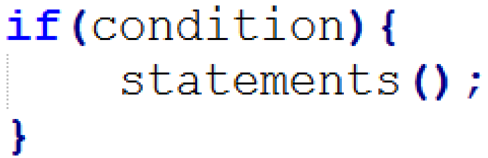
Selections and Loops

Selections

The one-way if Statements

- A one-way if statement executes an action if and only if the condition is true
- If the condition is false, nothing is done

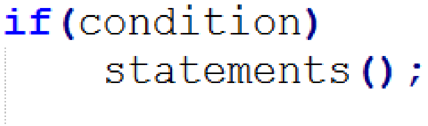
The syntax for a one-way if statement is:



```
if(condition){
      statements();
}
```

The one-way if Statements

- If the one-way if statement contains **just one** statement, the the block braces can be omitted as shown below.



```
if(condition)
    statements();
```

Selections and Loops

The one-way if Statements

Using one-way if statements only, write a Java program which takes an integer as input and prints out if the number is odd or even

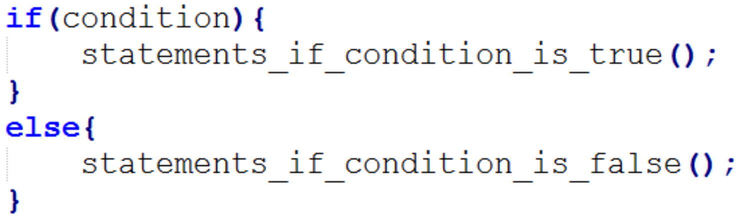
Selections and Loops

Selections

The two-way if-else Statements

- An if-else statement decides the execution path based on whether the condition is true or false.

The syntax for a two-way if-else statement is:



```
if(condition){
    statements_if_condition_is_true();
}
else{
    statements_if_condition_is_false();
}
```

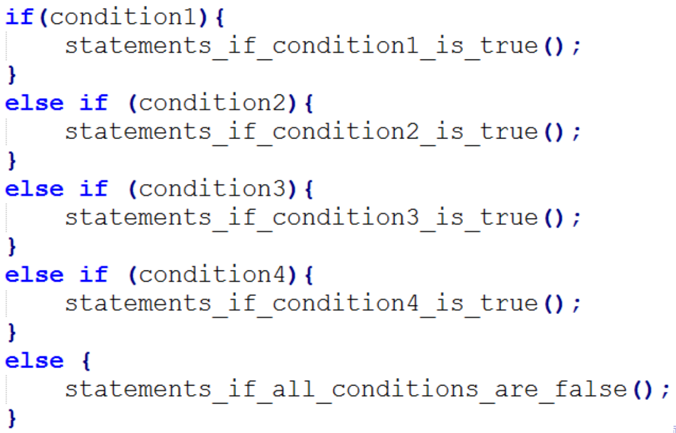
The two-way if-else Statements

Using the two-way if-else statements, write a Java program which takes an integer as input and prints out if the number is even or odd.

The Nested if and Multi-way if-else Statements

- An if statement can be inside another if statement to form a nested if statement.
- The inner if statement is said to be nested inside the outer if statement.
- There is no limit to the depth of the nesting.
- The multi-way if-else statements are used when there alternative conditions to be tested.

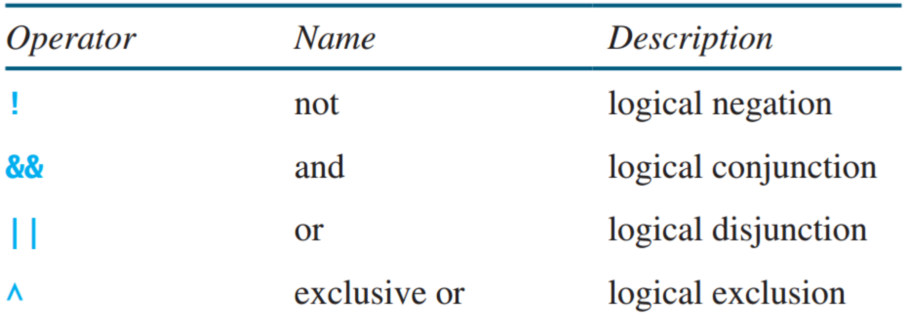
The Nested if and Multi-way if-else Statements



```
if(condition1){
    statements_if_condition1_is_true();
}
else if (condition2){
    statements_if_condition2_is_true();
}
else if (condition3){
    statements_if_condition3_is_true();
}
else if (condition4){
    statements_if_condition4_is_true();
}
else {
    statements_if_all_conditions_are_false();
}
```

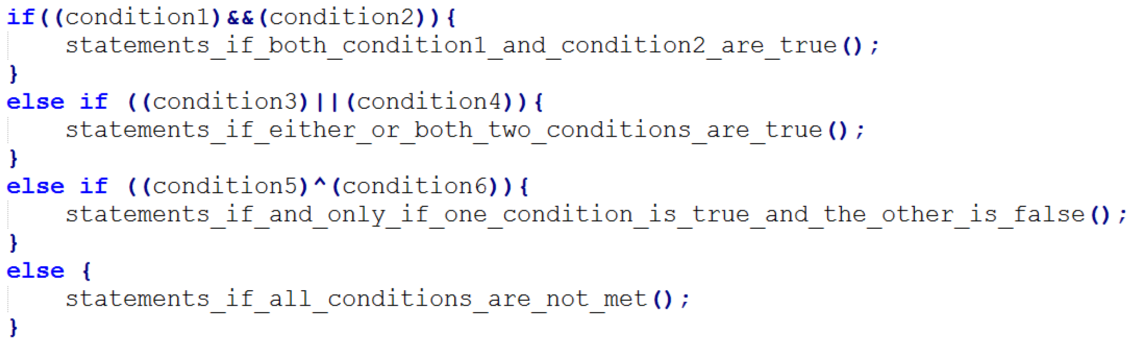
Logical Operators

- The logical operators shown below can be used to create compound conditions in if and if-else statements



Operator	*Name*	*Description*
!	not	logical negation
&&	and	logical conjunction
||	or	logical disjunction
^	exclusive or	logical exclusion

Multi-way if-else Statements with Logical Operators



```
if((condition1)&&(condition2)){
    statements_if_both_condition1_and_condition2_are_true();
}
else if ((condition3)||(condition4)){
    statements_if_either_or_both_two_conditions_are_true();
}
else if ((condition5)^(condition6)){
    statements_if_and_only_if_one_condition_is_true_and_the_other_is_false();
}
else {
    statements_if_all_conditions_are_not_met();
}
```

The Nested if and Multi-way if-else Statements

Write a Java program which takes marks as a double and returns the class range of the student as follows:

- mark $\geq$ 80.0 “First class”
- 70.0 $\leq$ mark $<$ 80.0 “Second class Upper”
- 60.0 $\leq$ mark $<$ 70.0 “Second class lower”
- 50.0 $\leq$ mark $<$ 60.0 “Pass”
- mark $<$ 50.0 “Fail”

Selections and Loops

switch Statements

A switch statement executes statements based on the value of a variable or an expression.

Selections and Loops

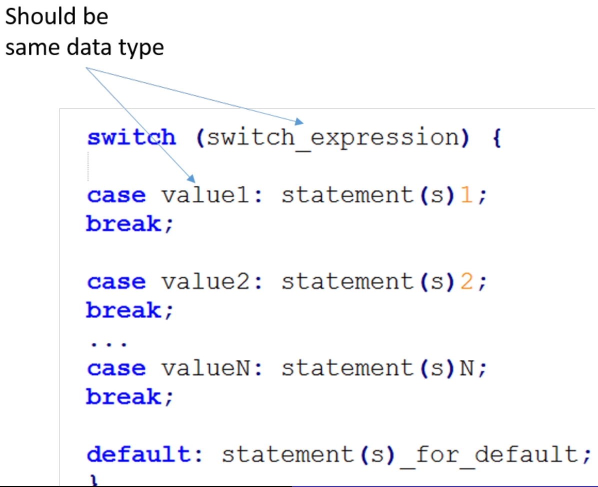
switch Statements

The syntax for the switch statement is as follows:

```
switch (switch_expression) {

case value1: statement(s)1;
break;

case value2: statement(s)2;
break;
...
case valueN: statement(s)N;
break;

default: statement(s)_for_default;
}
```

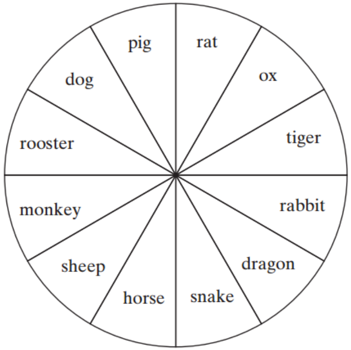
Selections and Loops

switch Statements



```
switch (switch_expression) {

case value1: statement(s)1;
break;

case value2: statement(s)2;
break;
...
case valueN: statement(s)N;
break;

default: statement(s)_for_default;
}
```

switch Statements



$$\texttt{year \% 12} = \begin{cases} \text{0: monkey} \\ \text{1: rooster} \\ \text{2: dog} \\ \text{3: pig} \\ \text{4: rat} \\ \text{5: ox} \\ \text{6: tiger} \\ \text{7: rabbit} \\ \text{8: dragon} \\ \text{9: snake} \\ \text{10: horse} \\ \text{11: sheep} \end{cases}$$

Given the Chinese Zodiac sign above, write a Java program (using switch statements) which accepts a user input and returns the Zodiac animal for the respective year.

if-else Statements vs switch Statements

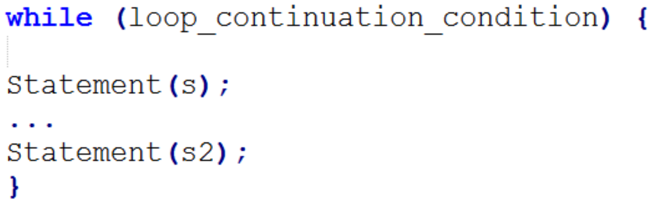
- Deciding on whether to use if-else statements or a switch statement is based on readability and the expression that the statement is testing.
- While an if-else statement can test expressions based on ranges of values or conditions, a switch statement tests expressions based only on a single integer, enumerated value, or String object.

Loops

- Loops can be used to tell a program to execute statements repeatedly.
- Java provides three types of loop statements:
  1. while loops
  2. do-while loops
  3. for loops

The while Loop

- A while loop executes statements repeatedly while the condition is true.
- The syntax for the while loop is as follows:

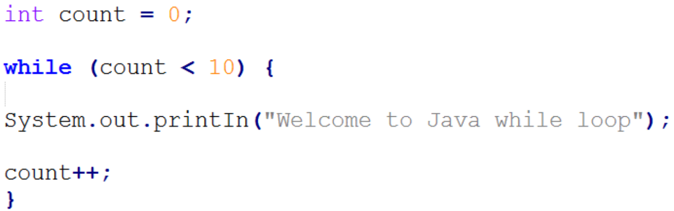


```
while (loop_continuation_condition) {

Statement(s);
...
Statement(s2);
}
```

- The loop continuation condition is checked first before the statements are executed

The while Loop

The following is an example of a while loop which prints 10 lines of the statement "Welcome to Java while loop"



```
int count = 0;

while (count < 10) {

System.out.printIn("Welcome to Java while loop");

count++;
}
```

Using the while loop, write a Java code which prints out the first 5 multiples of 2.

The do-while Loop

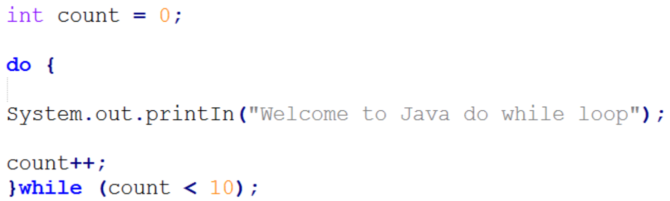
- The do-while loop is the same as a while loop except that it executes the loop body first and then checks the loop continuation condition
- The syntax for the do-while loop is as follows:

```
do {

Statement(s);
...
Statement(s2);

} while (loop-continuation-condition);
```

Selections and Loops

The do-while Loop

The following is an example of a do-while loop which prints 10 lines of the statement "Welcome to Java do-while loop"



```
int count = 0;

do {

System.out.printIn("Welcome to Java do while loop");

count++;
}while (count < 10);
```

Using the do-while loop, write a Java code which finds the sum of the first 8 integers.

Selections and Loops

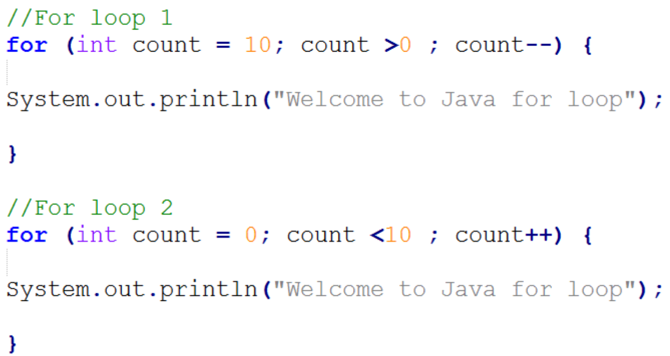
The for Loop

- A for loop performs an initial action once, then repeatedly executes the statements in the loop body, and performs an action after an iteration when the loop-continuation-condition evaluates to true
- The syntax for the for loop is as follows:

```
for (initial_action; loop_continuation_condition; action_after_each_iteration) {

Statement(s);

}
```

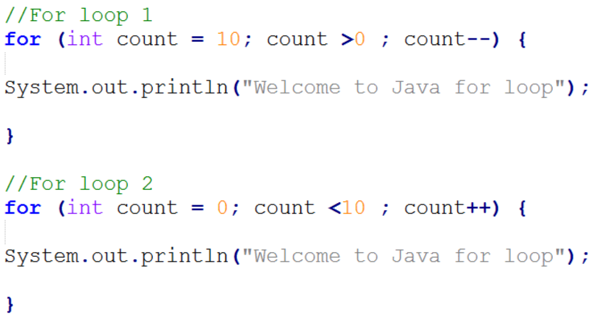
The for Loop

The following are two examples of for loops which both print 10 lines of the statement "Welcome to Java for loop"



```
//For loop 1
for (int count = 10; count >0 ; count--) {

System.out.println("Welcome to Java for loop");

}

//For loop 2
for (int count = 0; count <10 ; count++) {

System.out.println("Welcome to Java for loop");

}
```

The for Loop



```
//For loop 1
for (int count = 10; count >0 ; count--) {

System.out.println("Welcome to Java for loop");

}

//For loop 2
for (int count = 0; count <10 ; count++) {

System.out.println("Welcome to Java for loop");

}
```

Using the for loop, write a Java code which finds the sum of the first 10 integers.

Which Loop to Use?

- You can use a for loop, a while loop, or a do-while loop, whichever is convenient.
- The three forms of loop statements - while, do-while, and for - are expressively equivalent
  - That is, you can write a loop in any of these three forms

Note:

- The while loop and for loop are called **pretest** loops because the continuation condition is checked before the loop body is executed.
- The do-while loop is called a **posttest** loop because the condition is checked after the loop body is executed

Nested Loops

- A loop can be nested inside another loop
- Nested loops consist of an outer loop and one or more inner loops.
- Each time the outer loop is repeated, the inner loops are re-entered, and started anew.

Practice when you get back to your halls/hostels/homes.

References

- Y. Daniel Liang (2014), Introduction to Java Programming, Comprehensive Version, 10th Edition
- Cay S. Horstmann (2012), Java Concepts: Early Objects, 7th Edition.
- Deitel, P. and Deitel, H. (2015) Java How to Program, 10th Edition (Early Objects)