ICS 102 Lab #04: Selection Statements

Objectives:

1. Learn to use relational and equality operators
2. Learn to use **boolean** operators ! (not), && (logical AND), and || (logical OR)
3. Learn to use if and switch statements:

* Two-way selection (if-else statement)
* One-way selection (if-statement)
* Multi-way selection with else option (if - else if – else)
* Multi-way selection with no else option (if – else if – else if)
* Nested if statements
* switch statement with default option
* switch statement with no default option

In Java, conditions are **boolean** expressions that evaluate to either **true** or **false**. The following relational and equality operators are used to form simple conditions:

|  |  |
| --- | --- |
| Operator | Meaning |
| **==** | equals |
| **!=** | Not equal |
| **>** | Greater than |
| **>=** | Greater or equal |
| **<** | less |
| **<=** | Less or equal |

Examples:

**x >= y**

**2\*a\*b != k \* m**

**Math.pow(b, 2) – 4\*a\*c > 0**

The **boolean** operators **&&** (and) and **| |** (or) are used to form complex conditions:

**x > = y && y <= z**

**grade < 0.0 || grade > 100.0**

**ch1 >= 'a' && ch1 <= 'z' || ch1 >= 'A' && ch1 <= 'Z'**

**Operator Precedence**

In Java, mathematical and boolean expressions are evaluated according to the following precedence and associativity rules:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Operators | Order of Evaluation of operands with same precedence  (Associativity) |
| Higher Priority  Low Priority | 1 | (*expression*) and method calls | Left to right |
| 2 | var++ var-- | Right to left |
| 3 | ! , unary +, unary – , ++var, **--**var, (type) | Right to left |
| 4 | \*, /, % | Left to right |
| 5 | Binary +, binary - , String concatenation + | Left to right |
| 6 | **<, <=, >=, >** | Left to right |
| 7 | **==, !=** | Left to right |
| 8 | && | Left to right |
| 9 | | | | Left to right |
| 10 | **= , \*= , /=, %= , +=, -=** | Right to left |

Note: Parentheses are also used to group sub-expressions to force a different precedence; such parenthetical expressions can be nested and are evaluated from inner to outer.

Truth Tables

Let P and Q be **boolean** expressions

Truth table for the ! operator:

|  |  |
| --- | --- |
| P | ! P |
| true | false |
| false | true |

Truth table for the && operator:

|  |  |  |
| --- | --- | --- |
| P | Q | P && Q |
| true | true | true |
| true | false | false |
| false | true | false |
| false | false | false |

Truth table for the | | operator:

|  |  |  |
| --- | --- | --- |
| P | Q | P | | Q |
| true | true | true |
| true | false | true |
| false | true | true |
| false | false | false |

Two-way selection (if-else statement)

An *if-else* statement is used to execute a statement or a compound-statement when a condition is ***true***; and another statement or compound-statement when that condition is ***false***.

|  |  |
| --- | --- |
| Single statements | Compound statements |
| if(condition)  statement1;  else  statement2; | if(condition)  compound\_statement1  else  compound\_statement2 |
| Execute ***statement1*** if *condition* is *true* else execute ***statement2*** | Execute ***compound\_statement1*** if *condition* is *true* else execute ***compound\_statement2***  Note:   * It is a syntax error to put a semicolon after the closing brace of ***compound\_statement1*** * A compound statement may contain   zero or more statements |
|  |  |
| Example: Finding max of two numbers  if(x >= y)  max = x;  else  max = y; | Example:  char currencyType = scanner.nextLine().charAt(0);  if(currencyType = = 's' || currencyType = = 'S'){  System.out.println("Enter positive amount ");  amount = scanner.nextDouble();  riyalBalance = riyalBalance + amount;  }  else  System.out.println("Wrong currency type"); |

One-way selection (if-statement)

Used to execute a *statement* or a *compound\_statement* when a condition is ***true***.

|  |  |
| --- | --- |
| Single statements | Compound statements |
| if(condition)  statement1; | if(condition)  compound\_statement1 |
| Execute ***statement1*** if *condition* is ***true*** | Execute ***compound\_statement1*** if *condition* is ***true*** |
|  |  |
| System.out.println(“Enter a number: ”);  double num = scanner.nextDouble( );  if (num > 0)  System.out.printf(“The square root is %f”,  Math.sqrt(num)); | System.out.println(“Enter a number: ”);  double num = scanner.nextDouble( );  if (num > 0) {  System.out.printf(“The square root is %f”,  Math.sqrt(num));  System.out.printf(“The natural logarithm is %f”,  Math.log(num));  } |
| Example: Finding max of three numbers  max = num1;  if(num2 > max)  max = num2;  if(num3 > max)  max = num3; | Example:  char currencyType = scanner.nextLine().charAt(0);  if(currencyType = = 's' || currencyType = = 'S'){  System.out.println("Enter positive amount ");  amount = scanner.nextDouble( );  riyalBalance = riyalBalance + amount;  } |

Multi-way selection with an else option (if- else if - else statement)

Used to execute the first *statement* or the first *compound\_statement* whose corresponding condition is ***true***. The statement in the else part is executed if each condition is ***false***.

|  |  |
| --- | --- |
| Single statements | Compound statements |
| if(condition1)  statement1;  else if(condition2)  statement2;  else if(condition3)  statement3;  .  .  .  else if(conditionM)  statementM;  else  statementN; | if(condition1)  compound\_statement1  else if(condition2)  compound\_statement2  else if(condition3)  compound\_statement3  .  .  .  else if(conditionM)  compound\_statementM  else  compound\_StatementN |
| Note: There may be one or more *else if* branches | Note:   * There may be one or more *else if* branches. * It is a syntax error to put a semicolon   after the closing brace of a compound statement in an ***if***branch. |
|  |  |

|  |
| --- |
| boolean validGrade = true;  double grade;  System.out.println("Enter grade");  grade = scanner.nextDouble();  if(grade < 0.0 || grade > 100.0)  validGrade = false;  else if(grade >= 85.0)  letterGrade = 'A';  else if(grade >= 75.0)  letterGrade = 'B';  else if(grade >= 65.0)  letterGrade = 'C';  else if(grade >= 45.0)  letterGrade = 'D';  else  letterGrade = 'F';  if(validGrade)  System.out.printf("The letter grade is %c", letterGrade);  else  System.out.println("Error: Invalid grade"); |

Multi-way selection without an else option (if - else if - else if statement)

Used to execute the first *statement* or *compound\_statement* whose corresponding condition is ***true***. No if-branch is executed if each condition is ***false***.

|  |  |
| --- | --- |
| Single statements | Compound statements |
| if(condition1)  statement1;  else if(condition2)  statement2;  else if(condition3)  statement3;  .  .  .  else if(conditionM)  statementM;  else if(conditionN)  statementN; | if(condition1)  compound\_statement1  else if(condition2)  compound\_statement2  else if(condition3)  compound\_statement3  .  .  .  else if(conditionM)  compound\_statementM  else if(conditionN)  compound\_StatementN |
| Note: There may be one or more *else if* branches | Note:   * There may be one or more *else if* branches. * It is a syntax error to put a semicolon   after the closing brace of a compound statement in an *if* branch (except the last branch). |
|  |  |
| Example:  if(octaneNumber < 91)  System.out.println("Do not use this gasoline");  else if(octaneNumber < 95)  System.out.println("You may use this gasoline"); | |

Nested if statements

The compound statement in an if-branch or an else-branch of an if-statement may contain one or more of any type of if-statement discussed in the previous pages.

Example:

if(grade < 0.0 || grade > 100.0)

System.out.println("Error: Invalid grade");

else{

if(grade >= 85.0)

letterGrade = 'A';

else if(grade >= 75.0)

letterGrade = 'B';

else if(grade >= 65.0)

letterGrade = 'C';

else if(grade >= 45.0)

letterGrade = 'D';

else

letterGrade = 'F';

System.out.printf("The letter grade is %c", letterGrade);

}

Nested if structures can be complicated:

|  |  |
| --- | --- |
| if(condition1){  statement1;  if(condition2)  statement2;  else  statement3;  statement4;  }  else{  if(condition3)  compound\_statementA  else if(condition4)  compound\_statementB  else  compound\_StatementC  compound\_StatementD  } |  |

**Note:** In a nested if statement, the last else is associated with the closest unpaired if, unless braces are used to alter the default pairing:

Example:

if(hours < 9)

if(distance > 500)

System.out.println("Type 01");

else

System.out.println("Type 02");

is equivalent to:

if(hours < 9){

if(distance > 500)

System.out.println("Type 01");

else

System.out.println("Type 02");

}

Switch statement

The ***if- else if - else*** statement is used in programming situations where one set of statements must be selected from many possible alternatives. The **switch** statement provides an alternative to this statement for cases that compare the value of a **char, byte, short,** **int,** or **String** expression to a specific **char, byte, short,** **int, or String** constant. [Note: Using **String** in switch expression is supported in Java 7 and above]

The general form of the switch statement is:

**switch(expression){ // char, byte, short, int, or String expression**

**case constant1: statementList1;**

**break;**

**case constant2: statementList2;**

**break;**

**.**

**.**

**.**

**case constantM: statementListM;**

**break;**

**default: statementListN;**

**}**

**Note:**

* The case constants must be distinct char, byte, short, int, or String constants; otherwise there is a syntax error.
* A *statementList* may contain zero or more semi-colon separated statements. It is not necessary for *statementList* to be a compound-statement.
* The **default** label together with its *statementList* may be missing.
* The **default** label need not be the last label.
* The **break** statement following a *statementList* may be missing.
* The *switch expression* is evaluated and then the *statementList* of the case value that equals to the *expression* is executed. If there is a **break** statement, control passes to the statement after the switch; otherwise, the following *statementLists* are executed until a **break** statement is encountered, control then passes to the statement after the switch statement.
* If switch *expression* is not equal to any case value, the *statementList* for the **default** label is executed, and if the default label is the last one or if its statement list is followed by a break statement, control passes to the statement after the switch statement.
* If switch *expression* is not equal to any case value and there is no **default** label, control passes to the statement after the switch without executing any switch *statementList*.

**switch example01:** display the name of a digit:

int digit;

System.out.println("Enter an integer digit: ");

digit = scanner.nextDouble();

switch(digit){

case 0: System.out.println("zero");

break;

case 1: System.out.println("one");

break;

case 2: System.out.println("two");

break;

case 3: System.out.println("three");

break;

case 4: System.out.println("four");

break;

case 5: System.out.println("five");

break;

case 6: System.out.println("six");

break;

case 7: System.out.println("seven");

break;

case 8: System.out.println("eight");

break;

case 9: System.out.println("nine");

break;

default: System.out.println("Error: Invalid digit");

}

**switch example02:** classify a character

char c1;

System.out.println("Enter an English alphabet: ");

c1 = scanner.nextLine().charAt(0);

if(c1 >= 'a' && c1 <= 'z' || c1 >= 'A' && c1 <= 'Z'){

switch(c1){

case 'a':

case 'A':

case 'e':

case 'E':

case 'i':

case 'I':

case 'o':

case 'O':

case 'u':

case 'U': System.out.println("You entered a vowel");

break;

default: System.out.println("You entered a consonant");

}

}else

System.out.println("You entered a non-English alphabet");

**switch example03:** Determine month number from month string:

// Note: Strings in switch condition are supported in Java 7 and above

String month;

int monthNumber;

System.out.println(“Enter a month name:”);

month = scanner.next( );

switch (month.toLowerCase()) {

case "january": monthNumber = 1;

break;

case "february": monthNumber = 2;

break;

case "march": monthNumber = 3;

break;

case "april": monthNumber = 4;

break;

case "may": monthNumber = 5;

break;

case "june": monthNumber = 6;

break;

case "july": monthNumber = 7;

break;

case "august": monthNumber = 8;

break;

case "september": monthNumber = 9;

break;

case "october": monthNumber = 10;

break;

case "november": monthNumber = 11;

break;

case "december": monthNumber = 12;

break;

default: monthNumber = 0;

break; // optional

}

if(monthNumber != 0)

System.out.printf(“The month number for %s is %d%n”,

month, monthNumber);

else

System.out.println(“Error: Invalid month name”);

**Task 01**: Write a Java program that prompts and reads two integer numbers. It then checks the numbers and prints one of the following messages accordingly:

|  |
| --- |
| You have entered two even numbers. |
| You have entered two odd numbers. |
| You have entered one even number and one odd number. |

Hint: Use the modulus operator (**%**) for checking the numbers.

**Task 02**: Write a Java program that reads a character from a text file **input.txt**. It then writes to a text file **output.txt** one of the following messages accordingly:

The character x is lowercase

The character x is uppercase

The character x is character digit

where **x** is the character read from input.txt

If the input file does not exist, write the following message on the output file:

**Error: The input file input.txt does not exist**

and then terminate your program.

**Hint:**

* Use some of the **Character** class methods introduced in ICS-102 Lab04.
* Use File I/O methods introduced in ICS-102 Lab05.
* Use the exists( ) method of the class File introduced in ICS-102 Lab05.

**Task 03**: Convert the program-fragment below to a complete Java program that uses **if-else if -else** statement instead of the **switch** statement:

System.out.println(“Enter the student letter grade (A, B, C, D, or F) : “);

grade = scanner.nextLine().charAt(0);

switch (grade) {

case ‘A’:

case ‘a’:

case ‘B’:

case ‘b’: System.out.println(“Good standing”);

break;

case ‘C’:

case ‘c’ System.out.println(“O.K.”);

break;

case ‘D’:

case ‘d’:

case ‘F’:

case ‘f’: System.out.println(“Poor, student is on probation”);

break;

default: System.out.println(“Invalid letter grade”);

}

**Task 04:** Write a Java program that displays the following menu:

1. Find area and perimeter of a square

2. Find area and perimeter of a rectangle

3. Find area and perimeter of a circle

The program then reads the menu choice and behaves as in the following table:

|  |  |
| --- | --- |
| Menu choice | Program behavior |
| Input other than 1, 2, 3 | The program displays the following error message :  **Error: Wrong menu choice** |
| 1 | The program prompts for and reads the length of a square. It then computes and displays the area and the perimeter of the square. |
| 2 | The program prompts for and reads the length and the width of a rectangle. It then computes and displays the area and the perimeter of the rectangle. |
| 3 | The program prompts for and reads the radius of a circle. It then computes and displays the area and the circumference of the circle. [ , |

**Note:**

* Assume that the values read by the program for options 1, 2, and 3 are in **centimeters**. For these options, your program must display appropriate units in the output.
* You can solve the problem by using if- or switch-statement.

**Task05:** Write a Java program that prompts for and reads two strings **str1** and **str2**. The program then displays one of the following messages accordingly:

**str1** and **str2** are equal

**str1** is greater than **str2**

**str1** is less than **str2**

where **str1** and **str2** are the strings read. Your comparison must be case-sensitive.

**Hint**: Use the String comparison methods introduced in ICS-102 Lab04.