Making Decisions with the if Statement (Outsource: 4-1 – 4-4)

Making a decision involves choosing between two alternative courses of action based on some value within a program. The value upon which a decision is made is always a boolean value, which is always one of two values – *true* or *false*.

One statement you can use to make a decision is the if statement. When the condition of the if statement is evaluated as true, the statements following the if condition are executed. If the condition is evaluated as false, nothing happens, and the statement is ignored. This is called a ONE-WAY SELECTION statement or SINGLE-ALTERNATIVE if because you only perform an action based on one alternative.

Example: To output whether or not a person can vote based on their age, you write the following program statement:

```
if (age > 17)
    System.out.println("At age " + age + ", you are old enough to vote.");
```

The if…else Statement (Outsource: 4-9 – 4-16)

The **if…else** statement is the TWO-WAY SELECTION statement or dual-alternative if. The if…else statement performs one action when the condition is true, and another action when the condition is false.

Example: Same as above, but now another statement is made when the condition is false.

```
if (age >= 18)
    System.out.println("At age " + age + ", you are old enough to vote.");
else
    System.out.println("At age " + age + ", you CANNOT vote.");
```

The if…else if…else Statement (Outsource: 4-20 – 4-23)

The **if…else if…else** statement is the MULTIPLE SELECTION statement.

```
if (age < 4)
    System.out.println("Toddler.");
else if (age < 13)
    System.out.println("Pre-teen.");
else if (age < 20)
    System.out.println("Teenager.");
else
    System.out.println(“Adult”);
```

You have become accustomed to using semicolons to end statements. However, using a semicolon at the end of an **if** statement can cause problems, as shown below.

```
if ( S = = 3);       // Don't Do This
{
    System.out.println("The value of S is " + s);
}
```

The statement in braces will execute in every case because the compiler interprets the semicolon as the end of the **if** statement.

Comparison Operators (Outsource: 4-4 – 4-5)

When you set up the condition after the if, it must be in parentheses and it must be a comparison of two values, using the relational operators:

less than	<
greater than	>
less than or equal to	<=
greater than or equal to	>=
not equal to	! =
equal to	= =

W A R N I N G ! ! ! DO NOT USE = for equal to. It is the assignment operator. Be sure to use = = for equal to.

AND and OR Operators (Outsource: 4-6 – 4-7)

For an alternative to nested if statements, you can use the **AND** operator within a boolean expression to determine whether two expressions are both true. The AND operator is written as two ampersands (**&&**). For example the following program segment does exactly the same thing as the **nested if** shown above:

```
if (itemsSold > 3 && totalValue > 1000)
    bonus = 50;
```

It is important to note that when you use the AND operator, you must include a complete boolean expression on each side of the && operator.

With the AND operator, both boolean expressions must be true before the action in the statement can occur.

You can use the **OR** operator, which is written as ||, when you want some action to occur even if only one of the two conditions is true. For example, a common use of the OR operator is to decide to take action whether a character variable is uppercase or lowercase, as in **if (selection**

== 'A' || selection = = 'a') … the subsequent action occurs whether the selection variable holds an uppercase or lowercase A.

The NOT operator (Outsource: 4-7)

You use the **NOT** operator, which is written as the exclamation point (**!**), to negate the results of any boolean expression. Any expression that evaluates as true becomes false when preceded by the NOT operator, and any false expression preceded by the NOT operator becomes true.

Compound Statements

Often there is more than one action to take following the evaluation of a boolean expression within an if statement. To execute more than one statement you use a pair of curly brackets to place the statements within a block. For example, the following program segment determines whether an employee has worked more than 40 hours in a single week, and if so, the program computes regular and overtime salary and prints the results.

```
if (hoursWorked > 40)
{
    regularPay = 40 * rate;
    overTimePay = (hours –  40) * 1.5 * rate;       // Time and a half for hours over 40
    System.out.println("Regular pay is " + regularPay);
    System.out.println("Overtime Pay is " + overTimePay);
}
```

Nested *if* and nested *if…else* Statements

Within an if or an else statement, you can code as many dependent statements as you need, including other if and else statements. When you place an if statement inside another if statement it is commonly referred to as a **nested if statement**. Nested if statements are particularly useful when two conditions must be met before some action is taken. The following program segment shows a nested if statement:

```
if (itemsSold > 3)
{
    if (totalValue > 1000)
        bonus = 50;
}
```

Lab 07 - ASSIGNMENT

Lab 07A - 60 points (if…else)

OBJECTIVE

WAP to input a person's age and output three statements that report whether that person can vote (age 18), drive (age 17), or legally consume alcoholic beverages (age 21).
.

FIELD SUMMARY

- **int age** – a person's age.

METHOD SUMMARY

- **main** – instantiate an object of this class. Make method calls to `input` and `output`.
- **input** - declare a `Scanner` object and read a person's age from the keyboard using an appropriate prompt.
- **output** – display the results on the console screen. Use a **series** of **`if`**…**`else`** statements to produce the appropriate output based on age. Note that driving, voting, and drinking are not related events.

SAMPLE KEYBOARD INPUT

Enter your age: **17**

SAMPLE OUTPUT

At age 17, a person is old enough to drive.
At age 17, a person is not old enough to vote.
At age 17, a person is not old enough to drink alcohol legally.

Lab 07B - 70 points (if…else if)

OBJECTIVE

WAP to accept as input a NAME and a NUMERICAL GRADE from the keyboard. Print the input values in a complete sentence. Also, print a congratulatory message if the grade is an A or a condolence message if the grade is an F. No message is to be given for B, C, or D grades

FIELD SUMMARY

- **String name** – a person's name.
- **int grade** – a person's grade (in some undisclosed class).

METHOD SUMMARY

- **main** – instantiate an object of this class. Make method calls to `input` and `output`.
- **input** - declare a `Scanner` object and read a student's name and grade from the keyboard.
- **output** – display the results on the console screen. Use an **`if…else if`** statement to display an appropriate message, if required.

SAMPLE KEYBOARD INPUT

Enter a name: **Sue**
Enter a grade: **95**

SAMPLE OUTPUT

Sue, You Made A Grade Of 95. Congratulations!

SAMPLE KEYBOARD INPUT

Enter a name: **Sam**
Enter a grade: **78**

SAMPLE OUTPUT

Sam, You Made A Grade Of 78.

SAMPLE KEYBOARD INPUT

Enter a name: **Jerry**
Enter a grade: **66**

SAMPLE OUTPUT

Jerry, You Made A Grade Of 66. Poor boy, study harder!

Lab 07C - 80 points (if…else if…else)

OBJECTIVE

Given the coordinates of two points on a graph, find the slope of the line that passes through the two points. Remember that the slope of the line can be positive, negative, undefined or zero. Read four numbers from the keyboard in the order x1, y1, x2 and y2. Test for undefined (*delta x* = 0) and zero slope (*delta y* = 0). If neither *delta x* nor *delta y* is equal to 0 then, and only then, calculate the actual slope.

FIELD SUMMARY

- **int x1** – x value of the first coordinate.
- **int y1** – y value of the first coordinate.
- **int x2** – x value of the second coordinate.
- **int y2** – y value of the second coordinate.

METHOD SUMMARY

- **main** – instantiate an object of this class. Make method calls to `input` and `output`.
- **input** - declare a `Scanner` object and read the values for the two coordinates (four values in all).
- **output** – display the results on the console screen. Use an **`if…else if..else`** statement to display the appropriate messages. Test for extreme conditions first (undefined and 0 slope). What's left is a valid slope.

SAMPLE KEYBOARD INPUT

Enter the coordinates for the first point: **5 13**
Enter the coordinate for the second point: **5 2**

SAMPLE OUTPUT

The slope of a line containing (5, 13) and (5, 2) is undefined.

SAMPLE KEYBOARD INPUT

Enter the coordinates for the first point: **7 6**
Enter the coordinate for the second point: **9 6**

SAMPLE OUTPUT

The slope of a line containing (7, 6) and (9, 6) is zero.

SAMPLE KEYBOARD INPUT
Enter the coordinates for the first point: 6 5 **Enter the coordinate for the second point: 9 3**
SAMPLE OUTPUT
The slope of a line containing (6, 5) and (9, 3) is -0.667
SAMPLE KEYBOARD INPUT
Enter the coordinates for the first point: 4 7 **Enter the coordinate for the second point: 11 12**
SAMPLE OUTPUT
The slope of a line containing (4, 7) and (11, 12) is 0.714.

Lab 07D - 90 points (if…else if)

OBJECTIVE

WAP that determines your weight on another planet. The program should ask for the user's weight on Earth, then present a menu of the other planets in our solar system. The user should choose one of the planets from the menu and use a series of **`if`**…**`else if`** statements to calculate the weight on the chosen planet. Use the following conversion factors for the other planets.

Planet	Multiply by	Planet	Multiply by
Mercury	0.37	Saturn	1.15
Venus	0.88	Uranus	1.15
Mars	0.38	Neptune	1.12
Jupiter	2.64	Pluto	0.04

SAMPLE KEYBOARD INPUT

How much do you weigh? **150**

1] Mercury
2] Venus
3] Mars
4] Jupiter
5] Saturn
6] Uranus
7] Neptune
8] Pluto
Select a Planet. **3**

SAMPLE OUTPUT

On Mars you would weigh 57 pounds.

Lab 07E - 100 points (if…else)

OBJECTIVE

WAP that inputs taxpayer information from a data file. Two files have been provided for testing purposes (**"johnson.dat"** and **"ackerman.dat"**). Run the program using one of the file names then change the file name to the other file and run it again. Your program should compute the individual income tax, displaying all the pertinent information as shown in the example on the next page. Use the following table of constant values to perform the necessary calculations. (Constants should be declared as **final** values).

TAX_RATE = 0.15
STANDARD_SINGLE_DEDUCTION = 4750.0
STANDARD_MARRIED_DEDUCTION = 9500.0
INDIVIDUAL_DEDUCTION = 3050.0

The following information will be read from the data file:

lastName	**Taxpayer's Last Name**
firstName	**Taxpayer's First Name**
SSN	**Taxpayer's Social Security Number**
filingStatus	**Married or Single. (0 = Single, 1 = Married)**
numDependents	**The Number of dependents**
adjustedGrossIncome	**Wages, salaries, tips, etc.**
taxWithheld	**Amount of income tax withheld**

Using the two tables shown above make the following calculations:

taxableIncome	**adjustedGrossIncome – deductions**
totalTax	**taxableIncome * TAX_RATE**
amountOwed	**totalTax – taxWithheld.**

- **deductions** is computed as the appropriate STANDARD deduction plus numDependents times INDIVIDUAL_DEDUCTION.
- If **amountOwed < 0** then it should be refunded (as a positive value)

SAMPLE DATA FILE INPUT – "johnson.dat"

```
Johnson
Lloyd
512-87-9002
0
1
35520
6397
```

SAMPLE OUTPUT– "johnson.dat"

```
Income Tax Statement for Lloyd Johnson
SSN: 512-87-9002
Filing Status: Single with 1 dependents.
Adjusted Gross Income:     $35,520.00
Income Tax for 2006:       4,158.00
Tax Refund:  2,239.00
```

SAMPLE DATA FILE INPUT – "ackerman.dat"

```
Ackerman
Robert
143-56-8394
 1
 4
78552
 7200
```

SAMPLE OUTPUT

```
Income Tax Statement for Robert Ackerman
SSN: 143-56-8394
Filing Status: Married with 4 dependents.
Adjusted Gross Income:     $78,552.00
Income Tax for 2006:       8,527.80
Taxes Due:                 1,327.80
```