UNIVERSITY OF MISSOURI-COLUMBIA
COLLEGE OF ENGINEERING
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

3330 – OBJECT-ORIENTED PROGRAMMING

Laboratory 5: Decision Structures

Due Date: Friday October 3rd, 2025, at 5:00 pm

(No late submissions will be accepted for this Lab)

1 Goal

In this lab you will know how the conditional statement is used to create a decision structure, which allows a program to have more than one path of execution. The conditional statement causes one or more statements to execute only when a Boolean expression is true.

2 Resources

- Lecture notes "Unit 3"

3 Directed Lab Work

3.1 What is a conditional statement?

A conditional statement is a construct that allows a program to perform different computations depending on a boolean condition. If the condition is true, the program performs one computation; otherwise, the condition is false, and the program performs another computation. The condition can be any boolean expression, for instance: `a > b` , `i - j == 1` and so on.

There are three types of conditional statements. We will give examples of all of them.

3.1.1 A single if statement

The simplest form of the conditional statement is the following:

```
if (condition) {
    // do something
}
```

In this case, if the condition is true, the action inside the code block is executed. Otherwise, the program skips the action. See the following example.

```
int age = ...; // it has a value
if (age < 3) {
    System.out.println("This person is too young");
}
```

In this example, if the age is less than 3, the program prints "This person is too young". Otherwise, it does nothing.

3.1.2 If-else statement

The if statement can also be extended to do something else if the condition is false.

```
if (condition) {
    // do something
} else {
    // do something else
}
```

In the above code, if the condition is "true", the first code block is executed. Otherwise, the second code block is executed.

In the example below, the program outputs different text depending on the value of num (even or odd). Note that a whole number is even if it can be divided evenly by 2; otherwise, it's odd.

```
int num = ...; // num is assigned some value
if (num % 2 == 0) {
    System.out.println("It's an even number");
} else {
    System.out.println("It's an odd number");
}
```

For example, if the value of num is 10, the program outputs "It's an even number". If the value is 11, it outputs "It's an odd number".

3.1.3 If-else-if statements

The most general form of the if-else statement consists of multiple conditions and else-branches.

```
if (condition0) {
    // do something
} else if (condition1) {
    // do something else 1
// ...
} else if (conditionN) {
    // do something else N
}
```

The following code outputs recommendations as to what type of computer you should buy depending on your budget.

```
long dollars = ...; // your budget
if (dollars < 1000) {
    System.out.println("Buy a laptop");
} else if (dollars < 2000) {
    System.out.println("Buy a personal computer");
} else if (dollars < 100000) {
    System.out.println("Buy a server");
} else {
    System.out.println("Buy a data center or a quantum computer");
}
```

This conditional statement has four branches: dollars < 1000, dollars < 2000, dollars < 100000 and dollars >= 100000. For example, if the value of dollars is 9000, it prints Buy a server.

3.2 Task 1

- Use IntelliJ IDE to create a new Java project named Task1.
- Create a new Java class and name it **EvenOdd**.
- In this class write a **main** method that reads an integer number, prints EVEN if it is even. Or prints ODD if it is odd.
- Run your program and make sure it prints the correct output. Don't forget to add comments.

3.3 Task 2

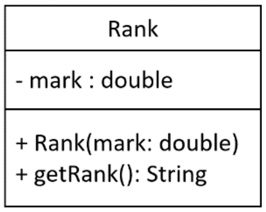
- Use IntelliJ IDE to create a new Java project named Task2.
- Create a new Java class and name it **IsLeapYear**.
- In this class write a **main** method that read a year number from the keyboard and print A LEAP YEAR if this year is a leap year or print NOT A LEAP YEAR otherwise.
- Hint: you can follow these steps to know whether a year is leap year or not. (1) If the year number is **not** divisible by 4 then it is **not** a leap year, (2) but if the year is divisible by 4 and not divisible by 100 then it is leap year, (3) also if the year is divisible by 4 and also divisible by 100, then the year is leap year if it is also divisible by 400 otherwise it is not leap year.
- Run your program and make sure it prints the correct output.
- Don't forget to add comments.

3.4 Task 3

- Once a student writes an exam, the instructor evaluates the students' performance and classify them into specific four categories based on their exam marks and the below mapping table.

Mark	Rank (Category)
< 50	Unacceptable
>= 50 and < 70	Below Expectations
>= 70 and < 90	Meets Expectations
>= 90	Exceeds Expectations

- Use IntelliJ IDE to create a new Java project named Task3, where you will develop a new class as shown in this UML diagram.



- The class constructor must initialize the class field with the given mark value (the argument).
- The getRank() method must use a nested if-else statements to examine the student's mark and returns the corresponding rank based on the above table conversion.
- In the same project file, create another class and name it RankDemo. Copy and paste the below code into RankDemo class. Do not change any statement or the logic of this given code.

```
import java.util.Scanner;

public class RankDemo {
    public static void main(String args []){
        double mark;  // To hold the student given mark

        // create a scanner object
        Scanner keyboard = new Scanner (System.in);

        // Prompt the user to enter the student's mark
        System.out.print("Enter the student's mark: ");
        mark = keyboard.nextDouble();

        // create the mark converter object using the rank class
        // initialized by the given student's mark
        Rank converter = new Rank(mark);

        // convert and display the mark and the corresponding rank category
        System.out.println("The mark " + mark + " is ranked as " + converter.getRank());
    }
}
```

- Run your program with multiple input values and make sure it prints the correct output.
- Don't forget to add comments.

3.5 Three keywords: switch, case and default

The switch statement provides a way to choose between multiple cases based on the value of a single variable. The variable in this statement can be char, byte, short, int, or string. The most general form of the switch statement looks as follows:

```
switch (variable) {
    case value1:
        // do something here
        break;
    case value2:
        // do something here
        break;

    //... other cases

    case valueN:
        // do something here
        break;
    default:
        // do something by default
}
```

The keywords switch and case are always required. The keywords break and default are optional.

Cases are evaluated sequentially. If a case is valid and it includes the break keyword, the switch construct is finished, and execution passes to the next statement. If a case doesn't include the break keyword, the following case(s) will be evaluated too. The default case is only evaluated if no case matches the variable value.

For Example: the following code output the name of some integers or the default string. It has three base cases and one default case.

```
int val = ...; // the variable is assigned some value
switch (val) {
     case 0:
        System.out.println("zero");
        break;
     case 1:
        System.out.println("one");
        break;
     case 2:
        System.out.println("two");
        break;
     default:
        System.out.println("The value is less than zero or greater than two");
}
```

If val equals 0, the code prints: `zero`

If val is 1, the code prints: `one`

if val is 10, the code prints: The value is less than zero or greater than two

> If you happen to forget to include break, the compiler will not consider it an error. Suppose we removed it from the second case (case 1) and assigned 1 to val. The program would print:
>
> one
>
> two

3.6 Task 4

- Use IntelliJ IDE to create a new Java project named Task4.
- Create a new Java class and name it **RankSwitchVersion**. In this class re-produce the code you developed in Task3 above using the switch statement instead of nested if-else statement.
- In the same project file, create another class and name it RankDemo. Copy and paste the code given for task3 into RankDemo class.
- Replace the statement

```
Rank converter = new Rank(mark);
```

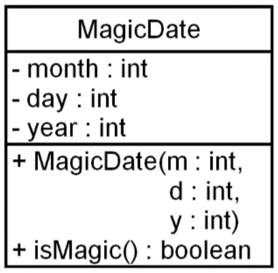
by the statement:

```
RankSwitchVersion converter = new RankSwitchVersion(mark);
```

- Do not change any other statements or the logic of this given code.
- Run your program with multiple input values and make sure it prints the correct output.
- Don't forget to add comments.

3.7 Task 5

- The date June 10, 1960, is special because when we write it in the following format, the month times the day equals the year.
  6/10/60



- Using IntelliJ IDE, create a new Java project called "**Task5**" that has two classes, a **MagicDate** class and a **MagicDatesDemo** class.
- Write the class MagicDate as it is shown by the UML diagram
- The class constructor should accept, as integers, values for a month, a day, and a year. The class should also have a method named isMagic that returns true if the date passed to the constructor is magic, or false otherwise.
- The main method in the MagicDatesDemo class asks the user to enter a month, a day, and a two-digit year as integers.
- The main method should create an instance of the MagicDate class to determine whether the date entered by the user is a magic date. If it is, the program should display a message saying the date is magic. Otherwise, it should display a message saying the date is not magic.
- Run your program and make sure it prints the correct output.
- Don't forget to add comments.

3.8 Task 6

- In this task you will be writing the quiz part of this lab. Use the Tests & Quizzes tab on Canvas to complete and submit answer for this quiz.
- The quiz includes one programming question to evaluate your understanding of this lab material. You will be given **30 minutes** to complete and submit your answer, please follow the following important instructions.
  - You are to complete this quiz independently and alone. Collaborating, discussing, or sharing of information during the quiz is not permitted.
  - Use the IntelliJ Idea IDE to create a new Java project file and named it yourUMId_Lab5_Quiz. yourUMId is the first part of your UM (University of Missouri) email address before the @ sign.
  - The quiz link will be inactive on **October 3 at 5:00 PM**. Note that, it is allowed to submit only once, so please review your code carefully before submitting your answer.
  - To start, click Final Assessment link. Once you click "Begin Assessment," you will have 30 minutes to complete this quiz.
  - Read the question and start to write your codes using the opened project file in the IntelliJ IDE.
  - Once you finished writing your program, export your project into yourUMId_Lab5_Quiz.zip.
  - To submit your answers, click the Browse button, navigate the file explorer to your exported zip file, click Upload button, and then click Submit for Grading button.

4 Hand In

- Create a new folder and name it yourUMId_Lab5, then copy the folders of projects from Task1 to Task5 in your new folder yourUMId_Lab5. yourUMId is the first part of your UM (University of Missouri) email address before the @ sign Zip yourUMId_Lab5 folder into yourUMId_Lab5.zip.

- **Please note that the <u>naming conventions and data values</u>, the types, data, and the submitted file structure are very important. The improper file structures, types and data or names will cause losing marks.**

- Submit your *yourUMId*_ Lab5.zip file into Canvas on **October 3, before 5:00 PM**.
- **No submissions will be accepted after October 3, 5:00 PM**