Assignment-3
Object Oriented Programming Lab
Java: Class and Object

1. Write a class named Employee that has the following fields:

   - name. The name field references a String object that holds the employee's name.
   - idNumber. The idNumber is an int variable that holds the employee's ID number.
   - department. The department field references a String object that holds the name of the department where the employee works.
   - position. The position field references a String object that holds the employee's job title.

   The class should have the following constructors:

   - A constructor that accepts the following values as arguments and assigns them to the appropriate fields: employee's name, employee's ID number, department, and position.
   - A constructor that accepts the following values as arguments and assigns them to the appropriate fields: employee's name and ID number. The department and position fields should be assigned an empty string ().
   - A no-arg constructor that assigns empty strings () to the name, department, and position fields, and 0 to the idNumber field.

   Write appropriate mutator methods that store values in these fields and accessor methods that return the values in these fields. Once you have written the class, write a separate program that creates three Employee objects to hold the following data: The program

Name	ID Number	Department	Position
Susan Meyers	47899	Accounting	Vice President
Mark Jones	39119	IT	Programmer
Joy Rogers	81774	Manufacturing	Engineer

   should store this data in the three objects and then display the data for each employee on the screen.

2. Write a class named Car that has the following fields:

   - yearModel. The yearModel field is an int that holds the car's year model.
   - make. The make field references a String object that holds the make of the car.
   - speed. The speed field is an int that holds the car's current speed.

   In addition, the class should have the following constructor and other methods.

   - Constructor. The constructor should accept the car's year model and make as arguments. These values should be assigned to the object's yearModel and make fields. The constructor should also assign 0 to the speed field.
   - Accessors. Appropriate accessor methods should get the values stored in an object's yearModel, make, and speed fields.

- accelerate. The accelerate method should add 5 to the speed field each time it is called.
- brake. The brake method should subtract 5 from the speed field each time it is called.

Demonstrate the class in a program that creates a Car object, and then calls the accelerate method five times. After each call to the accelerate method, get the current speed of the car and display it. Then call the brake method five times. After each call to the brake method, get the current speed of the car and display it.

3. Design a class **PersonalInformation** that holds the following personal data: name, address, age, and phone number. Write appropriate accessor and mutator methods. Demonstrate the class by writing a program that creates three instances of it. One instance should hold your information, and the other two should hold your friends' or family members' information.

4. Design a **Payroll** class that has fields for an employee's name, ID number, hourly pay rate, and number of hours worked. Write the appropriate accessor and mutator methods and a constructor that accepts the employee's name and ID number as arguments. The class should also have a method that returns the employee's gross pay, which is calculated as the number of hours worked multiplied by the hourly pay rate. Write a program that demonstrates the class by creating a Payroll object, then asking the user to enter the data for an employee. The program should display the amount of gross pay earned.

5. Design a **TestScores** class that has fields to hold three test scores. The class should have a constructor, accessor and mutator methods for the test score fields, and a method that returns the average of the test scores. Demonstrate the class by writing a separate program that creates an instance of the class. The program should ask the user to enter three test scores, which are stored in the **TestScores** object. Then the program should display the average of the scores, as reported by the **TestScores** object.

6. Write a **Circle** class that has the following fields:

   - radius: a double
   - PI: a final double initialized with the value 3.14159

   The class should have the following methods:

   - Constructor. Accepts the radius of the circle as an argument.
   - Constructor. A no-arg constructor that sets the radius field to 0.0.
   - setRadius. A mutator method for the radius field.
   - getRadius. An accessor method for the radius field.
   - getArea. Returns the area of the circle, which is calculated as area = PI * radius * radius
   - getDiameter. Returns the diameter of the circle, which is calculated as diameter = radius * 2
   - getCircumference. Returns the circumference of the circle, which is calculated ascircumference = 2 * PI * radius

   Write a program that demonstrates the Circle class by asking the user for the circle's radius, creating a Circle object, and then reporting the circle's area, diameter, and circumference.

7. Write a class named **MonthDays**. The class's constructor should accept two arguments:

   - An integer for the month (1 5 January, 2 February, etc.).
   - An integer for the year

   The class should have a method named getNumberOfDays that returns the number of days in the specified month. The method should use the following criteria to identify leap years:

   (a) Determine whether the year is divisible by 100. If it is, then it is a leap year if and if only it is divisible by 400. For example, 2000 is a leap year but 2100 is not.

   (b) If the year is not divisible by 100, then it is a leap year if and if only it is divisible by 4. For example, 2008 is a leap year but 2009 is not.

   Demonstrate the class in a program that asks the user to enter the month (letting the user enter an integer in the range of 1 through 12) and the year. The program should then display the number of days in that month.
   Here is a sample run of the program:
   Enter a month (1-12): 2 [Enter]
   Enter a year: 2008 [Enter]
   29 days

Note: Programming questions may be added, removed, or modified as per requirement