Exercise-3

1. Write a class that has three overloaded static methods for calculating the areas of the following geometric shapes:
   - circles
   - rectangles
   - cylinders

   Here are the formulas for calculating the area of the shapes:

   Area of a circle Area = pi*r^2

   where: pi is Math.PI and r is the circle's radius

   Area of a rectangle Area = Width x Length

   Area of a cylinder Area = pi*r^2*h

   where: n is Math.PI, r is the radius of the cylinder's base, and h is the cylinder's height

   Because the three methods are to be overloaded, they should each have the same name, but different parameter lists. Demonstrate the class in a complete program.
2. Make a LandTract class that has two fields, once for the tract's length and one for the width. The class should have a method that returns the tract's area, as well as an equals method and a toString method. Demonstrate the class the class in a program that asks the user to enter the dimensions for the two tracts of land. The program should display the area of each and indicate whether the tracts are of equal size.
3. Write a class named month. This class should have an int field named monthNumber that holds the number of months. For example, January would be 1. Feburary would be 2, and so forth. In addition, provide this following methods.
   - A no- arg constructor that sets the monthNumber field to 1.

- A constructor that accepts the number of the month as an argument. It should set the monthNumber field to the value passed as the argument. If a value less than 1 or greater than 12 is passed, the constructor should set monthNumber to 1.
- A Constructor that accepts the name of the month, such as "Janurary" or "February" as an argument. It should set the monthNUmber field to the correct corresponding value
- A setMonthNumber method that accepts an int argument, which is assigned to the monthNumber field. If a value less than 1 or greater than 12 is passed, the method should set monthNumber or 1.
- A getmonthNumber method that returns the value in the monthNumber field.
- A getMonthname method that returns the name of the month. For example, if the monthNumber field contains 1, then this method should return "january".
- A toString method that returns the same value as the getMonthName method.
- An equals method that accepts a month object as an argument. If the argument object holds the same data as the calling object, this method should return true. Otherwise it should return false.
- A greaterThan method that accepts a month object as an argument. If the calling object'smonthNumber field is greater than the argument's monthNumber field, this method should return true. Otherwise, it should return false.

- A lessThan method that accepts a Month object as an argument, If the calling object's monthNumber field is less than the argument's monthNumber field, this method should return true. Otherwise, it should return false.

4. For this assignment you will design a set of classes that work together to simulate a police officer issuing a parking ticket. You should design the following classes:
   - **TheParkedCarClass:** This class should simulate a parked car. The class's responsibilities are as follows:
     - To know the car's make, model, color, license number, and the number of minutes that the car has been parked.
   - **TheParkingMeterClass:** This class should simulate a parking meter. The class's only responsibility is as follows:
     - To know the number of minutes of parking time that has been purchased.
   - **TheParkingTicketClass:** This class should simulate a parking ticket. The class's responsibilities are as follows:
     -To report the make, model, color, and license number of the illegally parked car.
     -To report the amount of the fine, which is $25 for the first hour or part of an hour that the car is illegally parked, plus $10 for every additional hour or part of an hour that the car is illegally parked
     -To report the name and badge number of the police officer issuing the ticket.
   - **ThePoliceOfficerClass:** This class should simulate a police officer inspecting parked cars. The class's responsibilities are as follows:
     -To know the police officer's name and badge number.

-To examine a ParkedCar object and a ParkingMeter object, and determine whether the car’s time has expired.
-To issue a parking ticket (generate a ParkingTicket object). if the car's time has expired

Write a program that demonstrates how these classes collaborate.

5. For this assignment, you will design a set of classes that work together to simulate a car's fuel gauge and odometer. The classes you will design are the following:

The FuelGauge class: This class will simulate a fuel gauge. its responsibilities are as follows:

- To know the car’s current amount of fuel, in gallons.
- To report the car’s current amount of fuel, in gallons.
- To be able to increment the amount of fuel by 1 gallon. This simulates putting fuel in the car. (The car can hold a maximum of 15 gallons.)
- To be able to decrement the amount of fuel by 1 gallon, if the amount of fuel is greater than 0 gallons. This simulates burning fuel as the car runs.

The odometer class: This class will simulate the car’s odometer. its responsibilities are as follows:

- To know the car’s current mileage.
- To report the car’s current mileage.
- To be able to increment the current mileage by 1 mile. The maximum mileage the odometer can store is 999,999 miles. When this amount is exceeded, the odometer resets the current mileage to 0.
- To be able to work with a FuelGauge object. It should decrease the FuelGauge object’s current amount of fuel by 1 gallon for every 24 miles traveled. (the car’s fuel economy is 24 miles per gallon.)

Demonstrate the classes by creating instances of each. Simulate filling the car up with fuel, and then run a loop that

increments the odometer until the car runs out of fuel. During each loop iteration, print the car’s current mileage and amount of fuel.

6. Design a Geometry class with the following methods:
   - A static method that accepts the radius of a circle and returns the area of the circle.
     Use the following formula:
     Area=πr^2
     Use Math.PI for π and the radius of the circle for r.
   - A static method that accepts the length and width of a rectangle and returns the area of the rectangle. Use the following formula:
     Area = Length x Width
   - A static method that accepts the length of a triangle's base and the triangle's height. The method should return the area of the triangle. Use the following formula:
     Area = Base x Height x 0.5

The methods should display and error message if negative values are used for the circle's radius, the rectangle's length or width, or the triangle's base or height.

Next write a program to test the class, which displays the following menu and responds to the user's selection:

Geometry Calculator

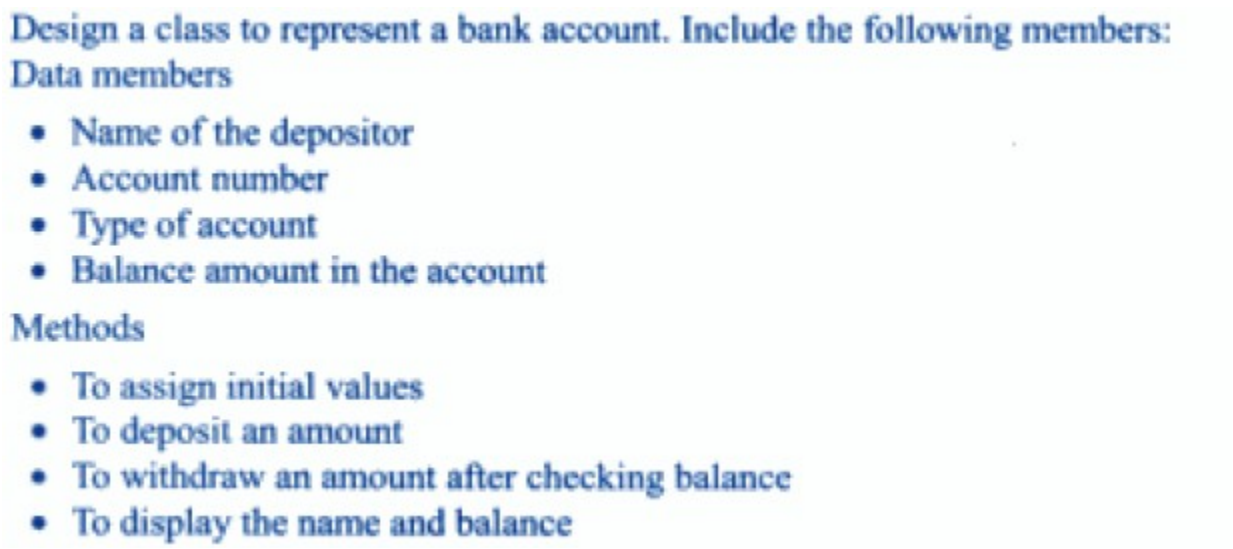
1. Calculate the Area of a Circle
2. Calculate the Area of a Rectangle
3. Calculate the Area of a Triangle

4. Quit

Entre your choice (1 - 4):

Display an error message if the user enters a number outside the range of 1 through 4 when selection an item from the menu.

7.



8. **Modify the program above to incorporate a constructor to provide initial value.**

9.

Assume that a bank maintains two kinds of account for its customers, one called savings account and the other current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed.

Create a class **Account** that stores customer name, account number and type of account. From this derive the classes **Curr-acct** and **Sav-acct** to make them more specific to their requirements. Include the necessary methods in order to achieve the following tasks:

(a) Accept deposit from a customer and update the balance.
(b) Display the balance.

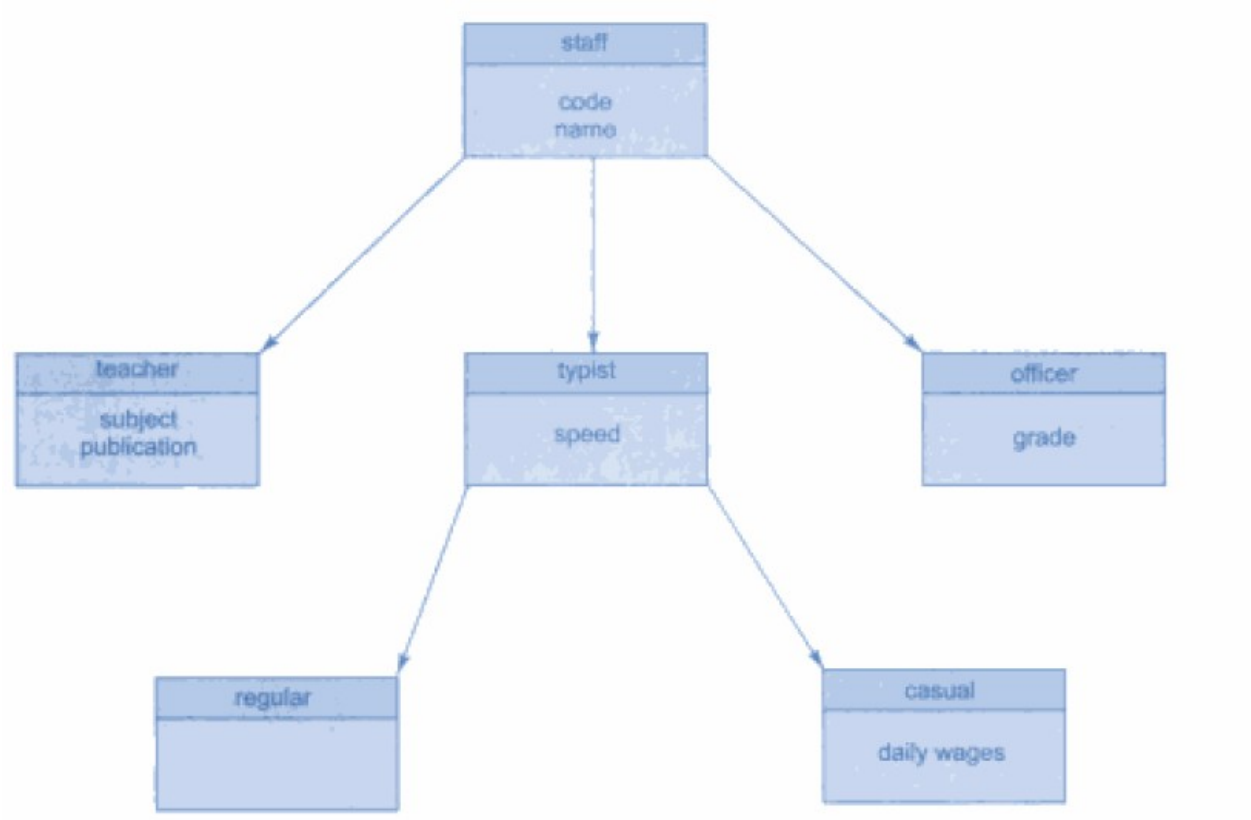
(c) Compute and deposit interest.
(d) Permit withdrawal and update the balance.
(e) Check for the minimum balance, impose penalty, if necessary, and update the balance.

Do not use any constructors. Use methods to initialize the class members.

10. **Modify the program above to include constructors for all the three classes.**

11.

An educational institution wishes to maintain a database of its employees. The database is divided into a number of classes whose hierarchical relationships are shown in Fig. 8.7. The figure also shows the minimum information required for each class. Specify all the classes and define methods to create the database and retrieve individual information as and when required.



12. Write a class named employee with 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 that store values in these fields and accessor methods that return the values in those fields. Once you have the class, write a separate program that creates three Employee objects to hold the following data:

Name	Id Number	Department	Position
Susan Meyers	47899	Accounting	vice Pres.
Mark Jones Joy	39119	IT	Programmer
Rogers	81774	Manufacturing	Engineer

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

13. Design a class that holds the following personal data: name, address, age, and phone number. Write appropriate accessor and mutator functions. 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.

14. 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 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.

15. Write a class named RetailItem that holds data about an item in a retail store. The class should have the following fields:

    Description: The description field references a String object that holds a brief description of the item.

    unitsOnHand: The unitsOnHand field is an int variable that holds the numbers of units currently in inventory.

    Price: The price filed is a double that holds the item's retail price.

Write a constructor that accepts arguments for each field, appropriate mutator methods that store values in these fields, and accessor methods that returns the values in these fields. Once you have written the class, write a separate program that creates three RetailItem objects and stores the following data in them:

	Description	Units on Hand	Price
Item #1	Jacket	12	59.95
Item #2	Designer Jeans	40	34.95
Item #3	Shirt	20	24.95

16. 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.

17. 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 as circumference= 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.

18. A bank charges $10 per month plus the following check fees for a commercial checking account:
$.10 each for fewer than 20 checks

$.08 each for 20 39 checks

$.06 each for 40 59 checks

$.04 each for 60 or more checks

The bank also charges an extra $15 if the balance of the account falls below $400 (before any check fees are applied). Write a program that asks for the beginning balance and the number of checks written. Compute and display the bank s service fees for the month.

19. Design a SavingsAccount class that stores a savings account's annual interest rate and balance. The class constructor should accept the amount of the savings account's starting balance. The class should also have methods for subtracting the amount of a withdrawal, adding the amount of a deposit, and adding the amount of monthly interest to the balance. The monthly interest rate is the annual interest rate divided by twelve. To add the monthly interest to the balance, multiply the monthly interest rate by the balance, and add the result to the balance.

Test the class in a program that calculates the balance of a savings account at the end of a period of time. It should ask the user for the annual interest rate, the starting balance, and the number of months that have passed since the account was established. A loop should then iterate once for every month, performing the following:

2. Ask the user for the amount deposited into the account during the month. Use the class method to add this amount to the account balance.

3. Ask the user for the amount withdrawn from the account during the month. Use the class method to subtract this amount from the account balance.

4. Use the class method to calculate the monthly interest.

After the last iteration, the program should display the ending balance, the total amount of deposits, the total amount of withdrawals, and the total interest earned.

20. The following table lists the freezing and boiling points of several substances.

Substance	Freezing Point	Boiling Point
Ethyl Alcohol	-173	172
Oxygen	-362	-306
Water	32	212

Design a class that stores a temperature in a temperature field and has the appropriate accessor andmutator methods for the field. In addition to appropriate constructors, the class should have the following methods:

- **isEthylFreezing:** This method should return the boolean value true if the temperature stored in the temperature field is at or below the freezing point of ethyl alcohol. Otherwise, the method should return false.
- **isEthylBoiling:** This method should return the boolean value true if the temperature stored in the temperature field is at or above the boiling point of ethyl alcohol. Otherwise, the method should return false.
- **isOxygenFreezing:** This method should return the boolean value true if the temperature stored in the temperature field is at or below the freezing point of oxygen. Otherwise, the method should return false.
- **isOxygenBoiling:** This method should return the boolean value true if the temperature stored in temperature field is at or above the boiling point of oxygen. Otherwise, the method should return false.
- **isWaterFreezing:** This method should return the boolean value true if the temperature stored in the temperature field is at or below the freezing point of water. Otherwise, the method should return false.
- **isWaterBoiling**: This method should return the boolean value true if the temperature stored in the temperature field is at or above the boiling point of water. Otherwise, the method should return false.

Write a program that demonstrates the class. The program should ask the user to enter a temperature, and then display a list of the substance that will freeze at that

temperature and those that will boil at that temperature. For example, if the temperature is -20 the class should report that water will freeze and oxygen will boil at that temperature.

21. Write a class named Coin. The Coin class should have the following field:

    - A String named sideUp. The sideUp field will hold either “heads” or “tails” indicating the side of the coin that is facing up.

    The Coin class should have the following methods:

    - A no-arg constructor that randomly determines the side of the coin that is facing up (“heads” or “tails”) and initializes the sideUp field accordingly.

    - A void method named toss that simulates the tossing of the coin. When the toss method is called, it randomly determines the side of the coin that is facing up (“heads” or “tails”) and sets the sideUp field accordingly.

    - A method named getSideUp that returns the value of the sideUp field.

    Write a program that demonstrates the Coin class. The program should create an instance of the class and display the side that is initially facing up. Then, use a loop to toss the coin 20 times. Each time the coin it tossed, display the side that is facing up. The program should keep count of the number of times heads is facing up and the number of times tails is facing up, and display those values after the loop finishes.