C S Patel Institute of Technology

U & P U Patel Computer Engineering Department

CE 251 - Java Programing

Practical Assignment

Assign Date: 20-09-2023

Deadline Date: 13-10-2023

1) Design a class named **Account** that contains:
    - A private **int** data field named **id** for the account (default 0).
    - A private **double** data field named **balance** for the account (default 0)
    - A private **double** data field named **annualInterestRate** that stores the current interest rate (default 0). Assume that all accounts have the same interest rate.
    - A private **Date** data field named **dateCreated** that stores the date when the account was created.
    - A no-arg constructor that creates a default account.
    - A constructor that creates an account with the specified id and initial balance.
    - The accessor and mutator methods for **id, balance** and **annualInterestRate**.
    - The accessor method for **dateCreated**.
    - A method named **getMonthlyInterestRate()** that returns the monthly interest rate.
    - A method named **getMonthlyInterest()** that returns the monthly interest.
    - A method named withdraw that withdraws a specified amount from the account.
    - A method named **deposit** that deposits a specified amount to the account.

2) In an n-sided regular polygon, all sides have the same length and all angles have the same degree (i.e., the polygon is both equilateral and equiangular). Design a class named **RegularPolygon** that contains:
    - A private **int** data field named n that defines the number of sides in the polygon with default value **3.**
    - A private **double** data field named **side** that stores the length of the side with default value **1**.
    - A private **double** data field named **x** that defines the x-coordinate of the polygon's center with default value **0**.
    - A private **double** data field named **y** that defines the y-coordinate of the polygon's center with default value **0**.
    - A no-arg constructor that creates a regular polygon with default values.
    - A constructor that creates a regular polygon with the specified number of sides and length of side, centered at **(0, 0)**.

- A constructor that creates a regular polygon with the specified number of sides, length of side, and x- and y-coordinates.
- The accessor and mutator methods for all data fields.
- The method **getPerimeter()** that returns the perimeter of the polygon.
- The method **getArea()** that returns the area of the polygon. The formula for computing the area of a regular polygon is:

$$\text{Area} = \frac{n \times s^2}{4 \times \tan\left(\frac{\pi}{n}\right)}.$$

3) Use the **Account** class created in Programming Exercise 1 to simulate an ATM machine.
   - Create 10 accounts in an array with id 0, 1, . . . , 9, and an initial balance of $100.
   - The system prompts the user to enter an id. If the id is entered incorrectly, ask the user to enter a correct id.
   - Once an id is accepted, the main menu is displayed.
   - You can enter choice 1 for viewing the current balance, 2 for withdrawing money, 3 for depositing money, and 4 for exiting the main menu.
   - Once the system starts, it will stop by entering number 99.

4) Create a class named **Stack**. Design a class named **Queue** for storing integers. Like a stack, a queue holds elements. In a stack, the elements are retrieved in a last-in first-out fashion. In a queue, the elements are retrieved in a first-in first-out fashion. The class contains:
   - An **int[]** data field named **elements** that stores the int values in the queue.
   - A data field named **size** that stores the number of elements in the queue.
   - A constructor that creates a **Queue** object with default capacity 8.
   - The method **enqueue(int v**) that adds v into the queue.
   - The method **dequeue()** that removes and returns the element from the queue.
   - The method **empty()** that returns true if the queue is empty.
   - The method **getSize()** that returns the size of the queue.

5) According to question no 1, the Account class was defined to model a bank account. An account has the properties account number, balance, annual interest rate, and date created, and methods to deposit and withdraw funds. Create two subclasses for checking and saving accounts. A checking account has an overdraft limit, but a savings account cannot be overdrawn.

6) Design a class named **Triangle** that extends **GeometricObject**.
   - The class contains: Three double data fields named side1, side2, and side3 with default values 1.0 to denote three sides of a triangle.
   - A no-arg constructor that creates a default triangle.
   - A constructor that creates a triangle with the specified side1, side2, and side3.
   - The accessor methods for all three data fields.
   - A method named **getArea()** that returns the area of this triangle.
   - A method named **getPerimeter()** that returns the perimeter of this triangle.
   - A method named **toString()** that returns a string description for the triangle. return **"Triangle: side1 = " + side1 + " side2 = " + side2 + " side3 = " + side3.**

7) Write a method public static int **readInFile(String line, File file)** that returns the position number of a line in the file filename or −1 if there is no such line or file.
   Assume that this file contains names of people with a name per line. Names (and hence lines) are listed in ascending alphabetical order in the file. We can not find the same line twice.

8) Write a program that will count the number of characters, words, and lines in a file. Words are separated by whitespace characters. The file name should be passed as a command-line argument.

9) Design an interface named **Colorable** with a void method named **howToColor()**. Every class of a colorable object must implement the **Colorable** interface. Design a class named **Square** that extends **GeometricObject** and implements **Colorable**. Implement **howToColor** to display the message **Color all four sides**. The **Square** class contains a data field **side** with getter and setter methods, and a constructor for constructing a **Square** with a specified side. The **Square** class has a private double data field named side with its getter and setter methods. It has a no-arg constructor to create a **Square** with side 0, and another constructor that creates a **Square** with the specified side

10) Define a class named **ComparableSquare** that extends **Square** and implements **Comparable**. Implement the **compareTo** method to compare the Squares on the basis of area. Write a **test** class to find the larger of two instances of ComparableSquareobjects.

11) Create a Triplet class that encapsulates three objects of the same data type in a given instance of Triplet.

12) Create an Association class that encapsulates two objects of different types. Similar to Exercise above, create a Transition class that does the same of Association class with three objects.

13) (Display nonduplicate names in ascending order) Given one or more text files, each representing a day’s attendance in a course and containing the names of the students who attended the course on that particular day, write a program that displays, in ascending order, the names of those students who have attended at least one day of the course. The text file(s) is/are passed as command-line argument(s)