EASTERN INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTING

AND INFORMATION TECHNOLOGY

Lab 4

Practice Assignment – Quarter 1, 2024-2025

Course Name: OOD

Course Code: CSE 203

Student's Full Name:

Student ID:

Inheritance

1. `Employee` and `ProductionWorker` Classes

Design a class named `Employee`. The class should keep the following information in fields:

- Employee name
- Employee number in the format XXX–L, where each X is a digit within the range 0–9 and the L is a letter within the range A–M.
- Hire date

Write one or more constructors and the appropriate accessor and mutator methods for the class.

Next, write a class named `ProductionWorker` that extends the `Employee` class. The `ProductionWorker` class should have fields to hold the following information:

- Shift (an integer)
- Hourly pay rate (a double)

The workday is divided into two shifts: day and night. The shift field will be an integer value representing the shift that the employee works. The day shift is shift 1 and the night shift is shift 2. Write one or more constructors and the appropriate accessor and mutator methods for the class. Demonstrate the classes by writing a program that uses a `ProductionWorker` object.

2. `ShiftSupervisor` Class

In a particular factory, a shift supervisor is a salaried employee who supervises a shift. In addi-tion to a salary, the shift supervisor earns a yearly bonus when his or her shift meets produc-tion goals. Design a ShiftSupervisor class that extends the Employee class you created in Programming Challenge 1. The ShiftSupervisor class should have a field that holds the annual salary and a field that holds the annual production bonus that a shift supervisor has earned. Write one or more constructors and the appropriate accessor and mutator methods for the class. Demonstrate the class by writing a program that uses a ShiftSupervisor object.

3. `TeamLeader` Class

In a particular factory, a team leader is an hourly paid production worker that leads a small team. In addition to hourly pay, team leaders earn a fixed monthly bonus. Team leaders are required to attend a minimum number of hours of training per year. Design a TeamLeader class that extends the ProductionWorker class you designed in Programming Challenge 1. The TeamLeader class should have fields for the monthly bonus amount, the required num-ber of training hours, and the number of training hours that the team leader has attended. Write one or more constructors and the appropriate accessor and mutator methods for the class. Demonstrate the class by writing a program that uses a TeamLeader object.

4. `Essay` Class

Design an Essay class that extends the GradedActivity class presented in this chapter. The Essay class should determine the grade a student receives for an essay. The student's essay score can be up to 100 and is determined in the following manner:

Grammar: 30 points

Spelling: 20 points

Correct length: 20 points

Content: 30 points

Demonstrate the class in a simple program.

5. Course Grades

In a course, a teacher gives the following tests and assignments:

- A lab activity that is observed by the teacher and assigned a numeric score.
- A pass/fail exam that has 10 questions. The minimum passing score is 70.
- An essay that is assigned a numeric score.
- A final exam that has 50 questions.

Write a class named CourseGrades. The class should have a GradedActivity array named grades as a field. The array should have four elements, one for each of the assignments previously described. The class should have the following methods:

setLab:	This method should accept a GradedActivity object as its argument. This object should already hold the student's score for the lab activity. Element 0 of the grades field should reference this object.
setPassFailExam:	This method should accept a PassFailExam object as its argument. This object should already hold the student's score for the pass/fail exam. Element 1 of the grades field should reference this object.
setEssay:	This method should accept an Essay object as its argument. (See Programming Challenge 4 for the Essay class. If you have not completed Programming Challenge 4, use a GradedActivity object instead.) This object should already hold the student's score for the essay. Element 2 of the grades field should reference this object.
setFinalExam:	This method should accept a FinalExam object as its argument. This object should already hold the student's score for the final exam. Element 3 of the grades field should reference this object.
toString:	This method should return a string that contains the numeric scores and grades for each element in the grades array.

Demonstrate the class in a program.

6. `Analyzable` Interface

Modify the CourseGrades class you created in Programming Challenge 5 so it implements the following interface:

```
public interface Analyzable
{
   double getAverage();
   GradedActivity getHighest();
   GradedActivity getLowest();
}
```

The getAverage method should return the average of the numeric scores stored in the grades array. The getHighest method should return a reference to the element of the grades array that has the highest numeric score. The getLowest method should return a reference to the element of the grades array that has the lowest numeric score. Demonstrate the new methods in a complete program.

7. `Person` and `Customer` Classes

Design a class named Person with fields for holding a person's name, address, and telephone number. Write one or more constructors and the appropriate mutator and accessor methods for the class's fields.

Next, design a class named Customer, which extends the Person class. The Customer class should have a field for a customer number and a boolean field indicating whether the customer wishes to be on a mailing list. Write one or more constructors and the appropriate mutator and accessor methods for the class's fields. Demonstrate an object of the Customer class in a simple program.

8. `PreferredCustomer` Class

A retail store has a preferred customer plan where customers can earn discounts on all their purchases. The amount of a customer's discount is determined by the amount of the cus-tomer's cumulative purchases in the store as follows:

- When a preferred customer spends $500, he or she gets a 5 percent discount on all future purchases.
- When a preferred customer spends $1,000, he or she gets a 6 percent discount on all future purchases.
- When a preferred customer spends $1,500, he or she gets a 7 percent discount on all future purchases.
- When a preferred customer spends $2,000 or more, he or she gets a 10 percent dis-count on all future purchases.

Design a class named PreferredCustomer, which extends the Customer class you created in Programming Challenge 7. The PreferredCustomer class should have fields for the amount of the customer's purchases and the customer's discount level. Write one or more construc-tors and the appropriate mutator and accessor methods for the class's fields. Demonstrate the class in a simple program.

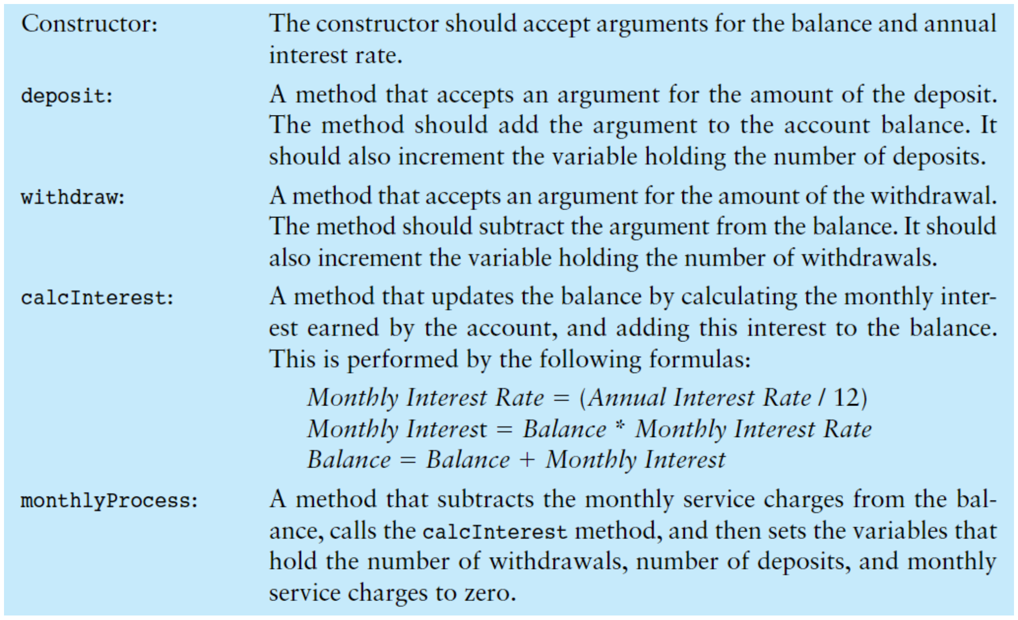
9. `BankAccount` and `SavingsAccount` Classes

Design an abstract class named BankAccount to hold the following data for a bank account:

- Balance
- Number of deposits this month
- Number of withdrawals

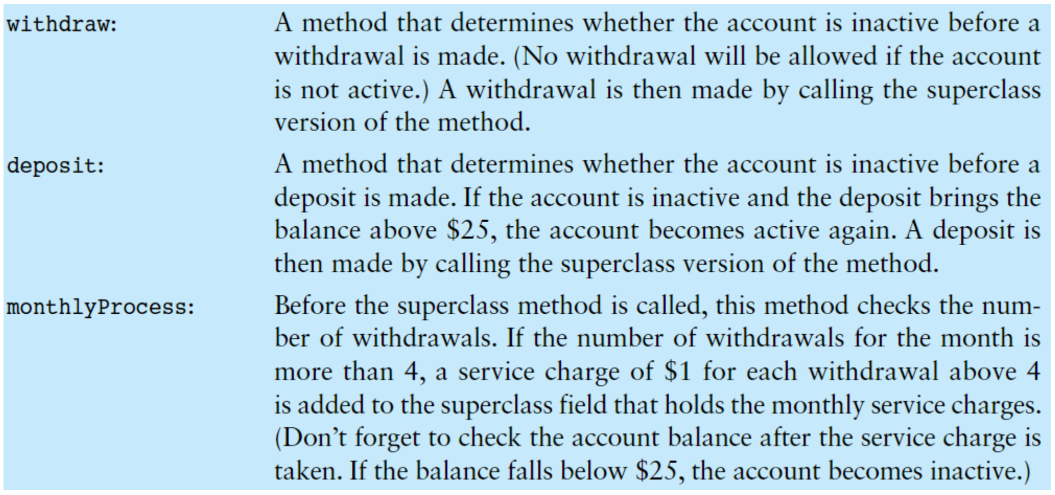
- Annual interest rate
- Monthly service charges

The class should have the following methods:



Constructor:	The constructor should accept arguments for the balance and annual interest rate.
`deposit:`	A method that accepts an argument for the amount of the deposit. The method should add the argument to the account balance. It should also increment the variable holding the number of deposits.
`withdraw:`	A method that accepts an argument for the amount of the withdrawal. The method should subtract the argument from the balance. It should also increment the variable holding the number of withdrawals.
`calcInterest:`	A method that updates the balance by calculating the monthly interest earned by the account, and adding this interest to the balance. This is performed by the following formulas: *Monthly Interest Rate* = (*Annual Interest Rate* / 12) *Monthly Interest* = *Balance* * *Monthly Interest Rate* *Balance* = *Balance* + *Monthly Interest*
`monthlyProcess:`	A method that subtracts the monthly service charges from the balance, calls the `calcInterest` method, and then sets the variables that hold the number of withdrawals, number of deposits, and monthly service charges to zero.

Next, design a `SavingsAccount` class that extends the `BankAccount` class. The `SavingsAccount` class should have a status field to represent an active or inactive account. If the balance of a savings account falls below $25, it becomes inactive. (The status field could be a boolean variable.) No more withdrawals may be made until the balance is raised above $25, at which time the account becomes active again. The savings account class should have the following methods:



`withdraw:`	A method that determines whether the account is inactive before a withdrawal is made. (No withdrawal will be allowed if the account is not active.) A withdrawal is then made by calling the superclass version of the method.
`deposit:`	A method that determines whether the account is inactive before a deposit is made. If the account is inactive and the deposit brings the balance above $25, the account becomes active again. A deposit is then made by calling the superclass version of the method.
`monthlyProcess:`	Before the superclass method is called, this method checks the number of withdrawals. If the number of withdrawals for the month is more than 4, a service charge of $1 for each withdrawal above 4 is added to the superclass field that holds the monthly service charges. (Don't forget to check the account balance after the service charge is taken. If the balance falls below $25, the account becomes inactive.)

10. `Ship`, `CruiseShip`, and `CargoShip` Classes

Design a `Ship` class that the following members:

- A field for the name of the ship (a `string`).
- A field for the year that the ship was built (a `string`).
- A constructor and appropriate accessors and mutators.

- A `toString` method that displays the ship's name and the year it was built.

Design a `CruiseShip` class that extends the `Ship` class. The `CruiseShip` class should have the following members:

- A field for the maximum number of passengers (an `int`).
- A constructor and appropriate accessors and mutators.
- A `toString` method that overrides the `toString` method in the base class. The `CruiseShip` class's `toString` method should display only the ship's name and the maximum number of passengers.

Design a `CargoShip` class that extends the `Ship` class. The `CargoShip` class should have the following members:

- A field for the cargo capacity in tonnage (an `int`).
- A constructor and appropriate accessors and mutators.
- A `toString` method that overrides the `toString` method in the base class. The `CargoShip` class's `toString` method should display only the ship's name and the ship's cargo capacity.

  Demonstrate the classes in a program that has a `Ship` array. Assign various `Ship`, `CruiseShip`, and `CargoShip` objects to the array elements. The program should then step through the array, calling each object's `toString` method.

Submit: Must include:

- a pdf file containing your information (student id, name), and images of the class diagrams, codes, answers
- and all source code files (if any)

in a zipped (.zip or *.rar) file to moodle