Bharatiya Vidya Bhavan's
Sardar Patel Institute of Technology
(Autonomous Institute Affiliated to Mumbai University)
[Knowledge is Nectar]

ACADEMIC YEAR 2023-24

Class: F.Y. B. Tech. Semester: II Course: PSOOP(Java)

Course In charge: Nikahat Mulla

List of Problems for practice-Set V-Composition/Aggregation/Abstraction Abstract Classes

1. Create a base class as a Vehicle. The Vehicle class has wheels and engine capacity as private data members and two abstract methods, spec() to set the values for data members and display_stats() to display the values assigned. Create classes LMV(Light Motor Vehicle), HMV(Heavy Motor Vehicle) derived from the Vehicle class. Include variables like speed, mileage and rpm in the derived classes and override the abstract methods in these classes. Also have constructor initializing the values to 0 as default. In main, create an array of references of the base class and set it to the objects of the derived classes.
   Now make a call to the various methods for these objects using the base class reference.

2. The Coworker class represents human resources data for a company. The class is used as a base class for various employees, blue-collar, whie-collar, and freelancers. The Coworker class has only a name, address of an employee, and the division where the employee works as private data members.
   Create a base class called CoWorker with the given data members. Declare abstract methods for computing income and displaying the worker's details and appropriate parameterized constructors. Create a derived class called Laborer which adds data members wages and hours. Class Laborer must override the income and display methods and have constructor, and other appropriate methods. Create another derived class called Employee with data member salary. Class Employee must override the income and display methods of CoWorker class and have constructor, and other appropriate methods. In main, create an array of CoWorker type and initialize them to different objects of derived classes by creating a menu to take appropriate details (Employee/Laborer). Call the display and income methods for these objects using the base class reference.

3. Design an abstract Investment class that includes a name attribute, a value attribute (double), and a getter method, getValue(). The Investment class, being abstract , cannot be instantiated. Design subclasses: Stocks, MutualFunds, RealEstate , and BankAccount .
   • The attributes of Stocks are name , pricePerShare , numberOfSharesOwned , and dividend (a percent of the investment paid annually).

• The attributes of MutualFunds are: name , pricePerShare , and numberOfSharesOwned. • The attributes of RealEstate are: name, addressOf Property , purchasePrice , and currentAssessedValue.

• BankAccount is an abstract class that extends Investment . The name field holds the bank's name. An additional attribute accountNumber (String) represents an account number. BankAccount has two subclasses: SavingsAccount and CheckingAccount .

• A SavingsAccount object has an annual interest rate paid quarterly. SavingsAccount has a method addInterest() that adjusts the balance of the account.

• A CheckingAccount is-a BankAccount with a minimum balance, a penalty if the balance goes below the minimum in any month, and an annual interest rate (paid monthly) on the money in excess of the minimum balance. Include method addInterest() , which adds one month's interest to the balance, and a method checkBalance(), which adjusts the balance if the balance falls below the minimum.

A portfolio is an array of Investment references. Implement a Portfolio class that also includes a getNetValue() method. This method returns the sum of the values of all investments referenced by portfolio. Interactively, create a portfolio with at least six investments, including stocks, mutual funds, real estate, and a bank account.

Display the data for each investment along with the net value of all investments.

4. A grocery store sells many different items. Construct an abstract class Item with attributes • String name ("apples" "soup" "candy bar")

• double unitPrice .

The methods of Item are getters and setters along with the requisite constructors. UnitItem and WeightItem are concrete classes that extend Item. An object belonging to UnitItem encapsulates a grocery item that is sold by the unit, such as a can of soup or a gallon of milk. The instance variable unitPrice (inherited from Item) stores the price of one item. UnitItem has an additional instance variable, amount, that holds the number of units of a particular item. UnitItem implements a method double cost() that returns the cost of amount units of an item.

WeightItem represents an item sold by weight, such as nuts, fruits, or vegetables. In this case, unitPrice represents the price per pound of an item. WeightItem has an additional instance variable, weight , that holds the number of pounds of some item.

WeightItem also implements a method

double cost()

WeightItem's implementation of cost() returns the total cost of weight pounds of the item. The weight of an item is set by placing the item on a scale. To simulate a scale, include a private helper method private double scale() that "weighs" the item and sets the weight fi eld. This is done by generating a random number, with two decimal places, between 0.01 and 4.00. The constructor uses this virtual "scale" to set the weight field. Both classes should include the appropriate constructors as well as getter and setter methods.

a. Design and implement Item, WeightItem , and UnitItem . Test your methods. b. A ShoppingCart class has an array of Item such that each array entry is a UnitItem or a WeightItem reference. Additionally, ShoppingCart implements a method

void checkout() that determines the total cost all items in the "cart," that is the array. A typical call to checkout() might produce the following interactive output:

Enter U or W or Return to end: U

Enter name: Soup

Number of Units: 2

Enter price per unit: 2.39

Cost is 4.78
Enter U or W: W
Enter name: Apples
Enter price per pound: 1.29
Weight is 2.8
Cost is 3.61
Enter U or W: W
Enter name: Green Beans
Enter price per pound: 1.19
Weight is 3.53
Cost is 4.2
Enter U or W: U
Enter name: Muffins
Number of Units: 6
Enter price per unit: .79
Cost is 4.74
Enter U or W:
Total cost: 17.33
Implement the ShoppingCart class. IncludeInclude a main(…) method that instantiates a ShoppingCart object and calls checkout().

5. A Company has a name, a product, and a list of employees. That is, a Company is composed of two String references (name and product), and an array of type Employee. Design a Company class. Methods should include constructors, getters, and setters, as well as methods that: • return a reference to an array of all the executives,
• return a reference to an array of all the managers who are not executives,
• return a reference to an array of all the employees who are neither managers nor executives, and
• return the sum of the salaries of all employees.
Include a main(...) method that tests the class. Your application should query the user for the name, product, and employee information. The user should indicate whether each employee is a manager or an executive, and it should include salary and other relevant information (Design Employee class according to this). The test class should display:
• the company name,
• company product,
• three lists of names and salaries:
• executives,
• managers (who are not executives), and
• employees (who are neither managers nor executives),
• the sum of the salaries for each list,
• and the sum of all the salaries of all employees.