**Practical 8: OO Design and Patterns**

Q1. **This question requires students to demonstrate understanding of the concept of *inheritance*, *composition*, *abstract classes* and *interfaces*.**

ABC Sdn. Bhd. has in employment a total of 50 full-time employees and 20 part-time staff. To inculcate the value of social responsibility among their employees, the management has decided to create 2 ways for full-time employees to contribute, i.e. either through contributing donations to or through doing volunteer work at one permanently adopted charity organization.

As a programmer working in the IT department, you are requested to write an application program to keep track of employees’ monthly payslip and social responsibility contribution details.

If a full-time employee chooses to donate, your program should prompt for the amount of donation the employee would like to contribute. Otherwise, employees will opt to volunteer as a “gotong-royong member”.

The calculation of monthly salary for employees are as follows:

* A full-time employee’s monthly salary is basic salary + allowance. If the employee has opted to donate to the charity fund, the donation amount should be deducted from the monthly salary.
* A part-time employee is paid by hourly rate \* hours worked. Assume that there is a standard hourly rate for all part-time employees.

Your program should display the total number of full-time employees who choose to contribute as volunteers and the total amount of donation contributed by the staff.

Design the classes required for this problem and draw a class diagram to show your design.

Note: To simplify the solution, ignore all other salary deductions (e.g. for EPF, PCB, etc.)

Answer

*+donate(amount: double): void*

*+volunteer(): void*

*<<interface>>*

***Contributor***

+CHARITY\_NAME: String

+ADDRESS: String

***Payslip***

-employee: Employee

-calendar: Calendar

+get and set methods

+getMonthName(): String

+getYear(): int

+getMonthAndYear(): String

+setCalendar(month: int, year: int): void

*+calculateSalary():double*

+toString(): String

**FullTimePayslip**

-monthlySalary: double

-contribution: char

-donationFund: double

-volunteerCount: int

+get and set methods

+donate(amount: double): void

+volunteer(): void

+calculateSalary(): double

+toString(): String

**PartTimePayslip**

-hourlyRate: double

-hours: int

+get and set methods

+calculateSalary(): double

+toString(): String

**Employee**

-id: int

-name: String

+get and set methods

+toString(): String

**FullTimeEmployee**

-basicSalary: double

-allowance: double

+get and set methods

+toString(): String

Q2. Develop an application for generating invoices. Some samples are shown below:

INVOICE

Invoice No: INV0001

Date: 29-Jun-2012

BILL TO: Cust. ID: C72635

Big Sdn. Bhd.

23 Jalan D9

42100

Selangor

-----------------------------------------------------------------------

Product No Description Qty Unit Price(RM) Line Total(RM)

1111 Toaster 3 90.00 270.00

2222 Electric Iron 1 120.00 120.00

-----------------------------------------------------------------------

Grand Total: RM390.00

INVOICE

Invoice No: INV0002

Date: 29-Jun-2012

BILL TO: Cust. ID: C58751

Tiger Sdn. Bhd.

25 Jalan 12

53000

Kuala Lumpur

-----------------------------------------------------------------------

Product No Description Qty Unit Price(RM) Line Total(RM)

2222 Electric Iron 2 120.00 240.00

3333 Air-conditioner 1 1200.00 1200.00

-----------------------------------------------------------------------

Grand Total: RM1440.00

INVOICE

Invoice No: INV0003

Date: 29-Jun-2012

BILL TO: Cust. ID: C72635

Big Sdn. Bhd.

23 Jalan D9

42100

Selangor

-----------------------------------------------------------------------

Product No Description Qty Unit Price(RM) Line Total(RM)

1111 Toaster 1 90.00 90.00

4444 Television 1 3000.00 3000.00

-----------------------------------------------------------------------

Grand Total: RM3090.00

(a) Determine all the necessary classes. Draw an UML class diagram showing the relationships between the classes (including their multiplicity). For each class, you only need to include the data members.

(b) Implement the classes from part (a).

Answer

**Address**

-street: String

-postcode: int

-state: double

**Customer**

-id String

-name: String

-address: Address

**Product**

-number: int

-description: String

-price: double

**InvoiceLine**

-product: Product

-quantity: int

-price: double

**Invoice**

-invoiceNumber: String

-date: Calendar

-customer: Customer

-lines: InvoiceLine[]

-totalLines: int

-nextInvoiceNumber: int

1 1

1 \*

1 \*

\* 1

Q3. Think of a drink vending machine.

1. What does the vending machine sell?
2. List the steps for using the vending machine.
3. What are the inputs and outputs of the vending machine?
4. Identify the two main components the vending machine.

IMPORTANT: Think using the object-oriented approach.

1. Identify the properties and operations of the 2 main components. Draw a class diagram to show the relationship between the vending machine and the 2 components, as well as the properties and operations of each class.
2. Write the classes for the vending machine. Include a driver program to test your classes.

Answer

1. A drink machine sells drinks such as coca-cola, sprite, orange and grape.
2. Steps for using the vending machine:
3. The machine displays the names and prices of the various drinks available in the machine. Drinks that are sold out will be indicated.
4. Customer selects the drink of his/her choice.
5. Machine displays the price of the drink selected.
6. Customer inserts money into the machine and the machine displays the balance to be inserted.
7. Once the total amount of money inserted into the machine is equal or exceeds the price of the drink, release the drink.
8. Inputs: Drink selection and the money to purchase the drink.

Output: The selected drink.

1. 2 main components of the vending machine:

* The cash register
* Various dispensers – normally there will be one or more dispenser per drink

**VendingMachine**

**Dispenser**

-cashRegister: CashRegister

-dispenserArr: Dispenser[]

-drink: string

-quantity: int

-price: double

\* 1

+ Constructors

+ get & set methods

+ sellProduct(Dispenser): void

+ Constructors  
+ get & Set methods

+ soldOut(): boolean

+ makeSale(): Boolean

+ compareTo(Object):int

1

1 1

**CashRegister**

-cashOnHand: double

+ CashRegister()

+getCashOnHand(): double

+acceptAmount(amount: double): void