Lehman College

OOP Practice Exercises

Problem 1. Geometry

Define a class structure that models a shape hierarchy.

- **Shape** – base class for any kind of shape, holds a list of **vertices**
    - **PlaneShape** – base class for all plane (2D) shapes, holds a list of **2D vertices** (holding **x** and **y**), implements **PerimeterMeasurable** and **AreaMeasurable** interfaces
        - **Triangle** – holds 3 vertices
        - **Rectangle** – holds 1 vertex, width, height
        - **Circle** – holds 1 vertex and radius
    - **SpaceShape** – base class for all three-dimensional shapes, holds a list of **3D vertices** (holding **x**, **y** and **z**), implements **AreaMeasurable** and **VolumeMeasurable** interfaces
        - **Square Pyramid** – holds 1 vertex (base center), base width, pyramid height
        - **Cuboid** – holds 1 vertex, width, height, depth
        - **Sphere** – holds 1 vertex and radius

A **vertex** is a point in 2D/3D space. The distance between two 2D vertices is calculated using the formula:

$$c = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$

Define the following interfaces:

- **PerimeterMeasurable** – holds **double getPerimeter()**
- **AreaMeasurable** – holds **double getArea()**
- **VolumeMeasurable** – holds **double getVolume()**

Design the class hierarchy using proper inheritance and code reusability through abstraction. Each shape should implement its respective interfaces with proper formulas.

Override **toString()** to return information about each shape (shape type, each vertex's coordinates, perimeter/area/volume). Create objects of different classes and add them to a **single** array. Iterate through the array and print information about each shape.

Problem 2. Shop

Design a class hierarchy that models a shop.

- **Product** – base class for all products, holds **name**, **price**, **quantity** and **age restriction** (can be **None**, **Teenager** or **Adult**). Implements the **Buyable** interface.
  - **FoodProduct** – implements the **Expirable** interface. Returns 70% of the price if the product expires in 15 days time.
  - **ElectonicsProduct** – base class for electronics, holds guarantee period
    - **Computer** – has a guarantee period of 24 months. Returns 95% of the price if the quantity is over 1000.
    - **Appliance** – has a guarantee period of 6 months. Returns 105% of the price if the quantity is less than 50.
- **Customer** – holds **name**, **age** and **balance**

Define **properties** (getters and setters) for each class. Validate the data and throw **exceptions** where necessary.

Define the following interfaces**:**

- **Buyable** – holds **double getPrice()**
- **Expirable** – holds **Date getExpirationDate()**

Create a static class **PurchaseManager**. The class should hold the **processPurchase(Product product, Customer customer)** method that handles purchases (takes money from customer, reduces product quantity by 1).The **PurchaseManager** should throw exceptions with descriptive messages in the following situations:

- If the product is out of stock (i.e. no quantity)
- If the product has expired
- If the buyer does not have enough money
- If the buyer does not have permission to purchase the given product

Catch any exceptions in your **main()** method and print their message. Create several products of different types and add them to a list. Filter the list using **lambda expressions** by:

- **Expirable** products and get the **name** of the first product with the soonest date of expiration
- All products with **adult age restriction** and **sort** them by **price** in ascending order