1. Design a class named BoardMarker with the following attributes:
    - Company (e.g., Dollar)
    - Color (e.g., black, red)
    - Refillable (Boolean – specifies if the marker can be refilled)
    - Ink status (Boolean – indicates whether the ink is empty)

    Implement appropriate getters and setters for these attributes. Additionally, include the following methods:
    - write() – This method should check if the ink is empty before allowing writing. If the ink is empty, display a message indicating that writing is not possible; otherwise, proceed with writing.
    - refill() – This method should verify if the marker is refillable. If it is, refill the ink and update the ink status; otherwise, display a message stating that the marker cannot be refilled.

    Demonstrate the functionality by creating 3 instances of BoardMarker, setting attribute values, and calling the methods to test writing and refilling operations for each of the 3 markers.

2. Create a Circle class with the following member variables: radius, a double representing the circle's radius, and pi, a double initialized with the value 3.14159. The class should include the following member functions: setRadius(), a mutator function to set the radius; getRadius(), an accessor function to retrieve the radius; getArea(), which returns the area of the circle using the formula area = pi * radius * radius; getDiameter(), which calculates and returns the diameter using diameter = radius * 2; and getCircumference(), which calculates and returns the circumference using circumference = 2 * pi * radius. Write a program that demonstrates the Circle class by prompting the user to input a radius, creating a Circle object, and displaying the calculated area, diameter, and circumference.

3. Create a class called water bottle. The water bottle has a company (made by), color and water capacity. The water capacity isstored in both liters(l) and milliliters(ml). Create variables and methods for your class. Methods should include getters and setters. Also create an additional method that updates the water capacity (both in l and ml) after asking user how much water a person has drank. Assume that the user always enters the amount in ml. Demonstrate the functionality of the water bottle in your main method.

4. Design a class for a StationeryShop that maintains a list of all items it sells, stored as an array of strings, along with a corresponding list of item prices, stored as an array of doubles. Create a menu-driven program that allows the shop owner to perform the following operations: add items and their prices, retrieve the list of items, edit item prices, and view all items along with their prices. Additionally, implement functionality to generate a receipt that the shopkeeper can share with customers. The receipt should be created only after the shopkeeper inputs the items purchased by the customer along with their quantities.

5. Design an Employee class that takes care of the not-so-fun part of earning money—taxes! Start by crafting a get_data function that collects the employee's name (because we need to know whose salary to slash), their monthly salary (so we can pretend like they're making money), and their tax percentage (because who doesn't love the feeling of money slipping through their fingers?). Next, add a Salary_after_tax function that dramatically deducts 2% tax from the salary—because why let them keep all of it, right? It returns the remaining salary, which will hopefully be enough to buy a cup of coffee (if they're lucky). But hold onto your hat! There's also an update_tax_percentage function that lets you increase the tax rate—because who doesn't enjoy surprising your employees with an unexpected increase in taxes (like 3% instead of 2%)? After that delightful change, the salary gets recalculated, and they get to see just how little they're left with. It's a perfect reminder that the only thing certain in life is taxes and less money!

6. Now, considering question 5, you are tasked with separating the class definition into a header file (Employee.h), then implementation file (Implement.cpp). Lastly, demonstrate it all by creating at least 3 instances in a separate file containing main function (main.cpp).