1. Create a class **BookStore** with the following attributes: [10]

Attributes:

- **bookTitle (String)** – Title of the book.
- **pricePerBook (double)** – Price per copy of the book.
- **stock (int)** – Number of available copies.
- **discount (double)** – Discount percentage (if applicable).

Constructor: A **parameterized constructor** to initialize all attributes with the following checks:

- If pricePerBook is less than or equal to 0, set it to **100.0** (default price).
- If bookTitle is empty (""), set bookTitle to **"Untitled"**.

Methods:

- **void sellBook(int quantity):**
  - If the requested quantity is greater than stock, display: **"Insufficient stock."**
  - Otherwise, reduce stock by the given quantity.
- **double calculateBill(int quantity):**
  - Calculate the total cost = quantity * pricePerBook.
  - Return the total cost.
- **void bookInfo():**
  - Display book details: title, price per book, available stock, and discount.

2. Design a class named **BookStoreApp** and inside the main method do the following: [10]

- Create **an instance** of the BookStore class using the parameterized constructor
- Use the **sellBook()** method to attempt to sell copies from both books
- Use the **calculateBill()** method for purchases.
- Call the **bookInfo()** method for both books.

1. Create a class **Flower** with the following attributes: [10]

Attributes:

- **flowerName (String)** – Name of the flower.
- **pricePerStem (double)** – Price per stem.
- **stemsInStock (int)** – Number of stems available.
- **daysSinceHarvest (int)** – Number of days since the flower was harvested.

Constructor: A **parameterized constructor** to initialize all attributes.

Methods:

- **void validateFlowerName():**
  - If flowerName is empty (""), set it to **"Unnamed Flower"**.
- **boolean isFresh():**
  - Returns true if daysSinceHarvest <= 5, otherwise false.
  - Display a message: ***"Flower is fresh."*** *or* ***"Flower is not fresh."***
- **double calculateCost(int quantity):**
  - Calculate total = quantity * pricePerStem.
  - If quantity >= 10, apply **10% bulk purchase discount**.
  - Return the total cost.
- **void displayFlowerInfo():**
  - Display all details: flower name, price per stem, stems in stock, days since harvest.

2. Design a class **FlowerShopApp** and inside the main method do the following: [10]

- Create **one Flower object** using the parameterized constructor
- Call **validateFlowerName()** for the object.
- Check freshness using **isFresh().**
- Use the **calculateCost()** method for a purchase of multiple stems.
- Call **displayFlowerInfo()**.

1. Create a class **MusicAlbum** with the following attributes: [10]

Attributes:

- **albumName (String)** – Name of the album.
- **artist (String)** – Artist of the album.
- **pricePerCopy (double)** – Price per copy of the album.
- **copiesAvailable (int)** – Number of available copies.
- **discount (double)** – Discount percentage (if applicable).

Constructor:

- A parameterized constructor to initialize **all attributes**.

Methods:

1. **void sellCopy(int quantity)**: If the requested quantity is **greater than available copies**, display: *"Not enough copies available."* and do not update stock. Otherwise, decrease the available copies.
2. **double calculateTotalCost(int quantity)**: If the customer buys **5 or more copies**, apply the discount to the total price. Return the final total cost after discount (if any).
3. **void albumInfo()**: Display all album details: album name, artist, price per copy, available copies, and discount.

2. Design a class named **MusicApp** and inside the **main** method do the following: [10]

a. Create **two** instances of the **MusicAlbum** class in Q1 using the parameterized constructor.
b. Use the **sellCopy()** method to attempt to sell copies from both albums (include a case where the order is larger than stock).
c. Use the **calculateTotalCost()** method for each purchase (ensure one qualifies for a discount).
d. Call the **albumInfo()** method to display details for both albums and **show the output of this method call.**

1. Design a class **OnlineCourse** with the following specifications: [10]

Attributes:

- **courseName (String)** – name of the course
- **capacity (int)** – maximum number of students allowed
- **enrolledStudents (int)** – number of students currently enrolled (initially 0)
- **pricePerStudent (double)** – enrollment fee per student

Constructor:

A parameterized constructor to initialize courseName , capacity and pricePerStudent. Set enrolledStudents as 0.

Methods:

int enrollStudent(int studentCount):

- Enroll the given number of students if seats are available; reject enrollment if seats are **insufficient**.
- If studentCount > 10, apply a **20% discount** on pricePerStudent for those enrollments.
- **Update** the number of enrolled students and print enrollment details with total cost.

int dropStudent(int studentCount):

- **Remove** the given number of students from the enrolled list. Reject if the number is invalid (greater than current enrollments or non-positive).
- Print confirmation after cancellation.

void displayCourse(): Display all details of the course including courseName, pricePerStudent, capacity, and enrolledStudents.

2. Design a new class **CourseApp**. Inside this class, define the **main** method and perform the following operations: [10]

a. Create **two** OnlineCourse objects of Q1 with different details.
b. Enroll students in both courses (including a case where more than 10 students are enrolled to apply the 20% discount).
c. Drop some students from both courses.
d. Display the details of both courses using the displayCourse() method after all operations.