Assignment #5 (CS210 – S24 – 10146)

## **Objective**:

* To understand and implement basic Java classes and inheritance by creating a hierarchy of geometric shapes.

# Coding #1: Geometric shapes (10 points)

### **Part A: Various geometric shape classes**

1. Create the Shape class:
   * Define the class as public.
   * Add a method double area() that returns 0.0 (to be overridden by subclasses).
   * Add a method double perimeter() that returns 0.0 (to be overridden by subclasses).
2. Create the Rectangle class that extends Shape:
   * Define instance variables for length and width.
   * Add a constructor that takes length and width as parameters and initializes them.
   * Override the double area() method to return the area of the rectangle.
   * Override the double perimeter() method to return the perimeter of the rectangle.
3. Create the Square class:
   * It’s your decision whether to extend Shape/Rectangle class and explain your reasoning.
4. Create the Circle class that extends Shape:
   * Define an instance variable for radius.
   * Add a constructor that takes radius as a parameter and initializes it.
   * Override the double area() method to return the area of the circle.
   * Override the double perimeter() method to return the circumference of the circle.

### **Part B: Unit tests**

1. Create a ShapeTest class to test the functionality:
   * Write test cases to validate the area() and perimeter() methods for each shape class.
   * Ensure that the test cases cover different scenarios, including edge cases.

# Coding #2: Library Management System (10 points)

## **Part A: Various Library Item Classes**

1. Create the LibraryItem class:
   * Define the class as public.
   * Declare instance variables for common attributes (e.g., title, author, year, ISBN).
   * Add a constructor to initialize the common attributes.
   * Create getter and setter methods for the instance variables.
   * Add a method void checkOut() that prints a message "Item checked out" (to be overridden by subclasses).
   * Add a method void returnItem() that prints a message "Item returned" (to be overridden by subclasses).
2. Create the Book class that extends LibraryItem:
   * Define instance variables for genre.
   * Add a constructor that takes title, author, year, ISBN, and genre as parameters and initializes them.
   * Override the checkOut() method to print a message "Book checked out".
   * Override the returnItem() method to print a message "Book returned".
3. Create the Magazine class that extends LibraryItem:
   * Define instance variables for issueNumber.
   * Add a constructor that takes title, author, year, ISBN, and issueNumber as parameters and initializes them.
   * Override the checkOut() method to print a message "Magazine checked out".
   * Override the returnItem() method to print a message "Magazine returned".
4. Create the DVD class that extends LibraryItem:
   * Define instance variables for duration.
   * Add a constructor that takes title, author, year, ISBN, and duration as parameters and initializes them.
   * Override the checkOut() method to print a message "DVD checked out".
   * Override the returnItem() method to print a message "DVD returned".

## **Part B: Main application class**

1. Create the LibraryApp class:
   * Define the class as public.
2. In the main method:
   * Instantiate at least one Book, one Magazine, and one DVD object with sample data.
   * Create an array to hold LibraryItem objects and add the instantiated objects to this array.
   * Iterate through the array, printing the details of each item, and call the checkOut() and returnItem() methods on each item.

## **Part C: Unit Tests**

1. Create a LibraryItemTest class to test the functionality:
   * Write test cases to validate the checkOut() and returnItem() methods for each library item class.
   * Write tests to validate various attributes.
   * Ensure that the test cases cover different scenarios, including edge cases.

### **Submission Details**

* **Code comments**: Ensure the code is well-commented.
* **Compilation and execution**: Verify that the code compiles and runs as expected.
* **Packaging:** package all project files in a zip file with your name as part of the file name.