**Assignment Description:-** Create a Java program that manages a list of students. Each student has a name, an age, and a grade. The program should allow adding students to the list, sorting the list based on the student's name, and handling exceptions related to invalid input.

**Requirements:**

1. Create a Student class with fields for name (String), age (int), and grade (double).
2. Implement a StudentManager class to manage the list of students. It should include methods for:

* Adding a new student to the list.
* Sorting the list of students based on their names.

1. Implement exception handling for the following scenarios:

* When adding a student, handle cases where the age is negative or the grade is not within the range of 0 to 100.
* When sorting the list, handle cases where the list is empty.

**Additional Tasks (Optional):**

* Implement a method to display the list of students.
* Use the Comparable interface to define the natural ordering of the Student class based on the student's name.
* Use the comparator interface to define the natural ordering of the Student class based on the student's grade
* Use the **compareTo** function for ordering of the Student class based on the student's age.

**Assignment Submission:**

* Create the Student and StudentManager classes according to the requirements.
* Implement exception handling for the specified scenarios.
* Optional: Implement additional methods and the Comparable interface for natural ordering.
* Write a brief explanation of your exception handling strategy.

**Assignment Description:** Create a Java program to manage a library system. The library system should handle books, patrons, and borrowing/returning books. The program should demonstrate the use of OOP principles, exception handling, collections, final, static, and constructor overloading.

**Requirements:**

1. Create a Book class with fields for **title, author, and isbn**. Include methods to display book details and to check if the book is available.
2. Create a Patron class with fields for **name, id, and booksBorrowed**. Include methods to borrow a book, return a book, and display patron details.
3. Create a Library class to manage the library system. It should include methods for:

* Adding a new book to the library.
* Registering a new patron.
* Handling book borrowing and returning.
* Displaying the list of books and patrons.

1. Implement exception handling for the following scenarios:

* When a patron tries to borrow more books than the allowed limit.
* When trying to add a duplicate book to the library.
* When trying to borrow a book that is not available.
* Any other relevant scenarios related to book borrowing and returning.

1. Use final and static keywords appropriately within your classes.
2. Implement constructor overloading in at least one of the classes to demonstrate different ways of creating objects.

**Assignment Submission:**

* Create the Book, Patron, and Library classes according to the requirements.
* Implement exception handling for the specified scenarios.
* Utilize the final and static keywords appropriately within your classes.
* Demonstrate constructor overloading in at least one of the classes.
* Write a brief explanation of your exception handling strategy and how you used final, static, and constructor overloading in your solution.

**Assignment 3:**

Create a Java program to model different types of **vehicles**. Implement a base class Vehicle and derive three classes **Car, Motorcycle, and Truck** from it. Each **vehicle** should have a method to calculate its fuel efficiency and a method to display vehicle details. Additionally, demonstrate polymorphism by creating an array of Vehicle objects and calling the fuel efficiency calculation method for each vehicle. Use Java collections to manage a list of vehicles, allowing the addition of new vehicles and the display of all vehicles in the list. Implement appropriate constructors and methods to achieve the desired functionality.

**Assignment 4:**

Develop a simple banking system in Java using OOP principles and Java collections. Create an abstract class Account with fields for **accountNumber, balance**, and **ownerName**. Derive two classes **SavingsAccount** and **CheckingAccount** from it. Implement a method to deposit money, a method to withdraw money, and a method to display the account details. Additionally, demonstrate inheritance and abstraction by designing the Account class as an abstract class and implementing the common functionality in it. Use Java collections to manage a list of accounts, allowing the addition of new accounts and the display of all accounts in the list. Use appropriate constructors and methods to handle account transactions and display account details.

Note :- Add testcases for all questions.