OOP in JAVA

1. **Create a Java class called BankAccount that represents a simple bank account. Include the following features:**

- Instance variables for account number, account holder's name, and account balance.
- Constructor to initialize the account details.
- Methods to deposit and withdraw funds from the account.
- A method to display the account details.
- Implement encapsulation and handle negative balances.

Write a Java program to create instances of BankAccount and perform deposit and withdrawal operations.

2. **Define a base class called Shape with methods for calculating area and perimeter. Create derived classes such as Rectangle and Circle that inherit from the Shape class and override the area and perimeter calculation methods.**

Write a Java program to create instances of different shapes (e.g., rectangles and circles) and demonstrate polymorphism by calculating their areas and perimeters using the same method names.

Collections in JAVA

3. . ArrayList Operations:

Write Java code to perform the following operations on an ArrayList:
- Add an element to the ArrayList.
- Remove an element from a specific index.
- Check if an element exists in the ArrayList.
- Retrieve the size of the ArrayList.
- Iterate through the ArrayList and print its elements.

4. HashMap Basics:

Create a HashMap in Java and perform the following operations:
- Add key-value pairs to the HashMap.
- Retrieve the value associated with a specific key.
- Check if a key exists in the HashMap.
- Iterate through the HashMap and print its key-value pairs.

5. LinkedList Operations:

Implement a singly linked list in Java and write code for the following operations:
- Add a node to the beginning of the list.
- Add a node to the end of the list.
- Delete a node with a specific value.
- Search for a node with a specific value.

6. Stack Implementation:

Implement a basic stack using an ArrayList in Java. Write code to perform the following stack operations:
- Push an element onto the stack.
- Pop an element from the stack.
- Check if the stack is empty.
- Peek at the top element of the stack without removing it.

7. Queue Implementation:

Implement a queue using a LinkedList in Java. Write code to perform the following queue operations:
- Enqueue (add) an element to the queue.
- Dequeue (remove) an element from the queue.
- Check if the queue is empty.
- Retrieve the size of the queue.

8. Sorting ArrayList:

Write Java code to sort an ArrayList of integers in ascending order using the `Collections.sort()` method.