Day-7 Assignment Questions:

1. All the banks operating in India are controlled by RBI. RBI has set a well defined guideline (e.g. minimum interest rate, minimum balance allowed, maximum withdrawal limit etc) which all banks must follow. For example, suppose RBI has set minimum interest rate applicable to a saving bank account to be 4% annually; however, banks are free to use 4% interest rate or to set any rates above it.
Write a program to implement bank functionality in the above scenario. Note: Create few classes namely Customer, Account, RBI (Base Class) and few derived classes (SBI, ICICI, PNB etc). Assume and implement required member variables and methods in each class.
Hint:

```
Class Customer
{
//Personal Details ...
// Few functions ...
}
Class Account
{
// Account Detail ...
// Few functions ...
}
Class RBI
{
Customer c; //hasA relationship
Account a; //hasA relationship
..
Public double GetInterestRate() { }
Public double GetWithdrawalLimit() { }
}
Class SBI extends public RBI
{
//Use RBI functionality or define own functionality.
}
Class ICICI extends public RBI
{
//Use RBI functionality or define own functionality.
}
```

2. Design a Payment Gateway System using an interface named PaymentMethod with a method pay(double amount). Implement this interface in different classes like CreditCardPayment, DebitCardPayment, and UPIPayment. Write a main class where you can accept payment using different methods.

3. Create a Java application to manage employees in a company. Define an abstract class Employee with a method calculateSalary(). Then create two subclasses FullTimeEmployee and PartTimeEmployee that override calculateSalary() method based on their working type. Demonstrate runtime polymorphism by calling calculateSalary() on different types of employees using the Employee reference.

4. Create a Java application to manage employees in a company. Define an Interface Employee with a method calculateSalary(). Then create two subclasses FullTimeEmployee and PartTimeEmployee that override calculateSalary() method based on their working type. Demonstrate runtime polymorphism by calling calculateSalary() on different types of employees using the Employee reference.

5. Develop a Java application for a Ticket Booking System that allows users to book tickets for different types of transportation modes such as *Bus*, *Train*, and *Flight*.
Define a common interface or abstract class `Ticket` with a method `bookTicket()` that each transportation mode must implement differently.
Create classes `BusTicket`, `TrainTicket`, and `FlightTicket` that extend the abstract class or implement the interface.
Demonstrate *runtime polymorphism* by calling the `bookTicket()` method using a reference of the base class/interface.