Q1.Write a simple Banking System program by using OOPs concept where you can get account Holder name balance etc?

Ans - import java.util.Scanner;

```
class BankAccount {
   private String accountHolder;
   private double balance;

   public BankAccount(String accountHolder, double initialBalance) {
      this.accountHolder = accountHolder;
      this.balance = initialBalance;
   }

   public void deposit(double amount) {
      if (amount > 0) {
         balance += amount;
         System.out.println("Deposited " + amount + " successfully.");
      } else {
         System.out.println("Invalid amount. Deposit failed.");
      }
   }

   public void withdraw(double amount) {
      if (amount > 0) {
         if (amount <= balance) {
            balance -= amount;
            System.out.println("Withdrew " + amount + " successfully.");
         } else {
            System.out.println("Insufficient balance. Withdrawal failed.");
         }
      } else {
         System.out.println("Invalid amount. Withdrawal failed.");
      }
   }

   public double getBalance() {
      return balance;
   }

   public String getAccountHolder() {
      return accountHolder;
   }
}
```

```
public class BankingSystem {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter account holder name: ");
        String accountHolder = scanner.nextLine();

        System.out.print("Enter initial balance: ");
        double initialBalance = scanner.nextDouble();

        BankAccount account = new BankAccount(accountHolder, initialBalance);

        System.out.println("Account holder: " + account.getAccountHolder());
        System.out.println("Initial balance: " + account.getBalance());

        System.out.print("Enter deposit amount: ");
        double depositAmount = scanner.nextDouble();
        account.deposit(depositAmount);

        System.out.print("Enter withdrawal amount: ");
        double withdrawalAmount = scanner.nextDouble();
        account.withdraw(withdrawalAmount);

        System.out.println("Final balance: " + account.getBalance());

        scanner.close();
    }
}
```

Q2. Write a Program where you inherit method from parent class and show method Overridden Concept?
Ans -

```
class Animal {
    public void makeSound() {
        System.out.println("Animal is making a sound");
    }
}

class Cat extends Animal {
    @Override
    public void makeSound() {
        System.out.println("Cat is meowing");
    }
}

class Dog extends Animal {
   @Override
   public void makeSound() {
      System.out.println("Dog is barking");
   }
}

public class AnimalDemo {
   public static void main(String[] args) {
      Animal animal = new Animal();
      animal.makeSound();

      Cat cat = new Cat();
      cat.makeSound();

      Dog dog = new Dog();
      dog.makeSound();
   }
}
```

Q3.Write a program to show run time polymorphism in java?

Ans -

```
      class Animal {
   public void makeSound() {
      System.out.println("Animal is making a sound");
   }
}

class Cat extends Animal {
   @Override
   public void makeSound() {
      System.out.println("Cat is meowing");
   }
}

class Dog extends Animal {
   @Override
   public void makeSound() {
      System.out.println("Dog is barking");
   }
}
```

```
public class RuntimePolymorphismDemo {
    public static void main(String[] args) {
        Animal animal1 = new Cat();
        Animal animal2 = new Dog();

        animal1.makeSound();
        animal2.makeSound();
    }
}
```

Q4.Write a program to show Compile time polymorphism in java?

Ans -

```
public class CompileTimePolymorphismDemo {
    public static int add(int num1, int num2) {
        return num1 + num2;
    }

    public static double add(double num1, double num2) {
        return num1 + num2;
    }

    public static String add(String str1, String str2) {
        return str1 + str2;
    }

    public static void main(String[] args) {
        int sum1 = add(5, 10);
        System.out.println("Sum of integers: " + sum1);

        double sum2 = add(2.5, 3.5);
        System.out.println("Sum of doubles: " + sum2);

        String result = add("Hello", "World");
        System.out.println("Concatenated string: " + result);
    }
}
```

Q5. Achieve loose coupling in java by using OOPs concept?

Ans - The loose coupling in Java shows how to achieve loose coupling in Java projects or programs. The more loosely coupled structures present in the project or program, the better it is. In loose coupling, a method or class is almost independent, and they have less depended on each other. In other words, the more knowledge one class or method has about another class or method, the more tightly coupled structure is developed. If the classes or methods know less about each other, the more loosely coupled structure comes into existence.

Q6. What is the benefit of encapsulation in java?
Ans - 1. Data Hiding
2. Increased Flexibility:
3.Reusability
4.testing code easy

Q7.Is java a t 100% Object oriented Programming language? If no why ?
Ans - Pure Object Oriented Language or Complete Object Oriented Language are Fully Object Oriented Language which supports or have features which treats everything inside program as objects. It doesn't support primitive datatype(like int, char, float, bool, etc.)

Q8.What are the advantages of abstraction in java?
Ans -

1. Avoids code duplication and increases reusability.
2. Helps to increase the security of an application or program as only essential details are provided to the user.
3. It improves the maintainability of the application.
4. It improves the modularity of the application.
5. The enhancement will become very easy because without affecting end-users we can able to perform any type of changes in our internal system.

Q9.What is an abstraction explained with an Example?

Ans - The interface provides complete abstraction i.e. it only provides method prototypes and not their implementation. An abstract class provides partial abstraction wherein at least one method should not be implemented.

Q10.What is the final class in Java?

Ans The final keyword in java is used to restrict the user, similarly, the final class means that the class cannot be extended. We can only create a final class if it is complete in nature, which means it cannot be an abstract class. All wrapper classes in Java are final classes, such as String, Integer, etc.