Assignment Questions 3

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

```
import java.util.Scanner;

class BankDetails {
   private String accno;
   private String name;
   private String acc_type;
   private long balance;
   Scanner sc = new Scanner(System.in);
   //method to open new account
   public void openAccount() {
      System.out.print("Enter Account No: ");
      accno = sc.next();
      System.out.print("Enter Account type: ");
      acc_type = sc.next();
      System.out.print("Enter Name: ");
      name = sc.next();
      System.out.print("Enter Balance: ");
      balance = sc.nextLong();
   }
   //method to display account details
   public void showAccount() {
      System.out.println("Name of account holder: " + name);
      System.out.println("Account no.: " + accno);
      System.out.println("Account type: " + acc_type);
      System.out.println("Balance: " + balance);
   }
   //method to deposit money
   public void deposit() {
      long amt;
      System.out.println("Enter the amount you want to deposit: ");
      amt = sc.nextLong();
      balance = balance + amt;
   }
```

```
   //method to withdraw money
   public void withdrawal() {
      long amt;
      System.out.println("Enter the amount you want to withdraw: ");
      amt = sc.nextLong();
      if (balance >= amt) {
         balance = balance - amt;
         System.out.println("Balance after withdrawal: " + balance);
      } else {
         System.out.println("Your balance is less than " + amt + "\tTransaction
failed...!!" );
      }
   }
   //method to search an account number
   public boolean search(String ac_no) {
      if (accno.equals(ac_no)) {
         showAccount();
         return (true);
      }
      return (false);
   }
}
public class BankingApp {
   public static void main(String arg[]) {
      Scanner sc = new Scanner(System.in);
      //create initial accounts
      System.out.print("How many number of customers do you want to input? ");
      int n = sc.nextInt();
      BankDetails C[] = new BankDetails[n];
      for (int i = 0; i < C.length; i++) {
         C[i] = new BankDetails();
         C[i].openAccount();
      }
      // loop runs until number 5 is not pressed to exit
      int ch;
      do {
         System.out.println("\n ***Banking System Application***");
```

```
        System.out.println("1. Display all account details \n 2. Search by Account
number\n 3. Deposit the amount \n 4. Withdraw the amount \n 5.Exit ");
        System.out.println("Enter your choice: ");
        ch = sc.nextInt();
           switch (ch) {
              case 1:
                 for (int i = 0; i < C.length; i++) {
                    C[i].showAccount();
                 }
                 break;
              case 2:
                 System.out.print("Enter account no. you want to search: ");
                 String ac_no = sc.next();
                 boolean found = false;
                 for (int i = 0; i < C.length; i++) {
                    found = C[i].search(ac_no);
                    if (found) {
                       break;
                    }
                 }
                 if (!found) {
                    System.out.println("Search failed! Account doesn't exist..!!");
                 }
                 break;
              case 3:
                 System.out.print("Enter Account no. : ");
                 ac_no = sc.next();
                 found = false;
                 for (int i = 0; i < C.length; i++) {
                    found = C[i].search(ac_no);
                    if (found) {
                       C[i].deposit();
                       break;
                    }
                 }
                 if (!found) {
                    System.out.println("Search failed! Account doesn't exist..!!");
```

```
                }
                break;
            case 4:
                System.out.print("Enter Account No : ");
                ac_no = sc.next();
                found = false;
                for (int i = 0; i < C.length; i++) {
                    found = C[i].search(ac_no);
                    if (found) {
                        C[i].withdrawal();
                        break;
                    }
                }
                if (!found) {
                    System.out.println("Search failed! Account doesn't exist..!!");
                }
                break;
            case 5:
                System.out.println("See you soon...");
                break;
            }
        }
        while (ch != 5);
    }
}
```

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

```
class Vehicle{
  void run(){System.out.println("Vehicle is running");}
}
//Creating a child class
```

```
class Bike extends Vehicle{
  public static void main(String args[]){
  //creating an instance of child class
  Bike obj = new Bike();
  //calling the method with child class instance
  obj.run();
  }
}
```

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

```
class Bike{
  void run(){System.out.println("running");}
}
class Splendor extends Bike{
  void run(){System.out.println("running safely with 60km");}

  public static void main(String args[]){
    Bike b = new Splendor();//upcasting
    b.run();
  }
}
```

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

```
public class MethodOverloading {

     // 1 parameter
    void show(int num1)
    {
       System.out.println("number 1 : " + num1);
    }

    // 2 parameter
    void show(int num1, int num2)
    {
       System.out.println("number 1 : " + num1
                    + "  number 2 : " + num2);
    }
```

```
    public static void main(String[] args)
    {
        MethodOverloading obj = new MethodOverloading();

         // 1st show function
        obj.show(3);

         // 2nd show function
        obj.show(4, 5);
    }
}
```

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

```
interface Parent
{
    void foo();
}

class A implements Parent
{
    // parameterized constructor
    A(int x, int y)
    {

    }

    public void foo()
    {
        System.out.println("In the foo method of class A.");
    }
}

class B implements Parent
{
    public void foo()
    {
```

```
        System.out.println("In the foo method of class B.");
    }
}

public class CouplingExample2
{

// main method
public static void main(String argvs[])
{
// creating an object of class B
B obj = new B();
obj.foo();
}
}
```

Q6. What is the benefit of encapsulation in java?

The main advantage of using encapsulation is the security of the data. Benefits of encapsulation include:

- Encapsulation protects an object from unwanted access by clients.
- Encapsulation allows access to a level without revealing the complex details below that level.
- It reduces human errors.
- Simplifies the maintenance of the application
- Makes the application easier to understand.

Q7. Is java a t 100% Object oriented Programming language? If no why ?

Java is not a pure object oriented language because it supports Primitive datatype such as int, byte, long? etc, to be used, which are not objects.

Java provides wrapper class for int, long, etc? But still int, long, float,etc? are not classes. Integer, Float, Long only classes. And There are some qualities to be satisfied for a programming language to be pure Object Oriented. They are:

1. Encapsulation/Data Hiding
2. Inheritance
3. Polymorphism

4. Abstraction
5. All predifined types are objects
6. All operations are performed by sending messages to objects
7. All user defined types are objects.

Q8.What are the advantages of abstraction in java?

Advantages of Abstraction in Java

Increasing Understandability of the Code

Abstraction is a technique that makes it easier to understand complex systems by converting the complete code into relatively smaller parts that can be managed & understood easily. Since we are hiding the specific details of how something works in the class, the code becomes simpler and easier to read.

Improves Flexibility

Through abstraction, programmers can alter a class's inner workings without affecting its users, thereby increasing the system's versatility. This streamlines the task of implementing changes to the system, ultimately boosting its overall Flexibility.

Promotes Code Reusability

By using abstraction, developers can create versatile classes that are applicable to a wide range of scenarios, paving the way for code reusability. This saves developers time and effort and prevents them from having to recreate similar code from scratch. By using abstraction, developers can create more flexible and adaptable code that can be easily modified and reused in new contexts just as explained in the above example.

Improves Modular Design

Abstraction encourages a modular design approach, which simplifies the process of modifying or updating a system without impacting its core functionality. This is especially beneficial for systems whose codes are vast and complex in nature but require frequent updates.

Q9.What is an abstraction explained with an Example?

As Java is an OOP language, abstraction can be seen as one of the important features and building blocks of the Java language. In Java, abstraction is implemented using an abstract class and interface.

```
//abstract class
abstract class Car{
   abstract void accelerate();
}
//concrete class
class Suzuki extends Car{
   void accelerate(){
      System.out.println("Suzuki::accelerate");

   }
}
class Main{
   public static void main(String args[]){
      Car obj = new Suzuki();    //Car object =>contents of Suzuki
      obj.accelerate();          //call the method
   }
}
```

Q10.What is the final class in Java?

As the name suggests, Final Class in Java uses the final keyword for its declaration. 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. Final class cannot be inherited by any subclass,