**Assignment Questions 3**

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

public class BankAccount {

    private String accountHolderName;

    private double balance;

    public BankAccount(String accountHolderName, double balance) {

        this.accountHolderName = accountHolderName;

        this.balance = balance;

    }

    public String getAccountHolderName() {

        return accountHolderName;

    }

    public double getBalance() {

        return balance;

    }

    public boolean deposit(double amount) {

        if (amount > 0) {

            balance += amount;

            return true;

        } else {

            System.out.println("Deposit amount in your account");

            return false;

        }

    }

    public boolean withdraw(double amount) {

        if (amount > 0) {

            if (balance >= amount) {

                balance -= amount;

                return true;

            } else {

                System.out.println("Bhai/Sis Balance daloo pase katem");

                return false;

            }

        } else {

            System.out.println("Withdrawal amount from account ");

            return false;

        }

    }

    public static void main(String[] args) {

        // Create a new bank account

        BankAccount account = new BankAccount(" Bhanu Pratap", 99199.00);

        // Get account holder name and balance

        System.out.println("Account holder name is- " + account.getAccountHolderName());

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

        // Deposit money

        account.deposit(500.0);

        System.out.println("After Updating Balance - " + account.getBalance());

        // Withdraw money

        account.withdraw(300.0);

        System.out.println(" After Updating Balance -" + account.getBalance());

    }

}

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

class Parent {

    void display() {

        System.out.println("Parents Method");

    }

}

class Child extends Parent {

    void display() {

        System.out.println("Child method");

    }

}

public class Main {

    public static void main(String[] ar) {

        Parent parent = new Parent();

        parent.display();

        Child child = new Child();

        child.display();

    }

}

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

class Parent {

    void display() {

        System.out.println("Parents Method");

    }

}

class Child extends Parent {

    void display() {

        System.out.println("Child Method");

    }

}

public class Main {

    public static void main(String[] args) {

        Parent bhanu = new Child();

        bhanu.display();

    }

}

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

class Papa {

    void display() {

        System.out.println("Papa ka Method");

    }

    void display(String message) {

        System.out.println("Papa Ka Message" + message);

    }

}

class Beta extends Papa {

    @Override

    void display() {

        System.out.println("Beta Ka method");

    }

}

public class Main {

    public static void main(String[] args) {

        Papa papa = new Papa();

        papa.display();

        Beta beta = new Beta();

        beta.display();

        Papa betaAsPapa = new Beta();

        betaAsPapa.display();

        betaAsPapa.display("Hello Ji Kase ho");

    }

}

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

Loose coupling in Java achieved using OOPs concepts by

Encapsulation- Hide implementation details and expose only necessary methods to interact with objects.

Abstraction-Define interfaces to communicate with objects, providing only essential functionality.

Inheritance-Use subclasses to extend and modify behavior without affecting the base class.

Polymorphism- Allow objects of different classes to be treated interchangeably based on shared behavior.

💡 **Q6. What is the benefit of encapsulation in java?**

The benefits of encapsulation in Java are:

1. Data hiding: Encapsulation protects the internal state of an object, preventing direct access to sensitive data.

2. Modularity: Encapsulation allows changes to an object's implementation without affecting other parts of the program.

3. Code flexibility: It simplifies debugging, maintenance, and refactoring as implementation details are hidden behind the interface.

4. Security: Encapsulation helps control access to data and provides better security for sensitive information.

💡 [Q7. Is](http://Q7.Is) **java a t 100% Object oriented Programming language? If no why ?**

No, Java is not a 100% Object-oriented Programming language due to the following reasons:

1. Primitive data types: Java includes primitive types like int, double, etc., which are not objects.

2. Static methods: Java supports static methods that belong to the class rather than objects.

3. Constructor calls: Constructor calls don't use inheritance, violating pure OOP principles.

4. Final classes: Java has final classes that cannot be extended, limiting the concept of inheritance.

💡 **Q8.What are the advantages of abstraction in java?**

1. Simplified complexity: Abstraction hides implementation details, making code more manageable.

2. Enhanced security: Sensitive data is concealed from external access.

3. Code reusability: Abstract classes and interfaces enable code sharing among multiple classes.

4. Flexibility: Implementations can change without affecting the code using the abstraction.

5. Ease of maintenance: Abstraction simplifies code updates and modifications.

💡 **Q9.What is an abstraction explained with an Example?**

Abstraction is a concept in Java that allows you to create a simplified representation of an object, focusing on essential characteristics while hiding implementation details.

Example: Shape (Abstraction)

- We define an abstract class "Shape" with an abstract method "calculateArea()".

- Concrete classes like "Circle" and "Rectangle" extend "Shape" and provide their specific implementations of "calculateArea()".

- Users can work with the abstract "Shape" class, unaware of the specific shape's implementation details.

💡 **Q10.What is the final class in Java?**

In Java, a final class is a class that cannot be subclassed or extended. Once a class is declared final, it cannot have subclasses, preventing further inheritance. The final keyword applied to a class ensures that its behavior cannot be modified or extended, providing rigidity to the class structure.