

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

```
package bank_account;
import java.util.Scanner;

public class BankAccount {

    private String name;
    private int accountNumber;
    private double balance;

    public BankAccount(String name, int accountNumber, double balance) {
        this.name = name;
        this.accountNumber = accountNumber;
        this.balance = balance;
    }

    public String getName() {
        return name;
    }

    public int getAccountNumber() {
        return accountNumber;
    }

    public double getBalance() {
        return balance;
    }

    public void deposit(double amount) {
        balance += amount;
    }

    public void withdraw(double amount) {
        if (balance < amount) {
            throw new IllegalArgumentException("Insufficient funds");
        }

        balance -= amount;
    }

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

        System.out.println("Enter account holder's name: ");
        String name = scanner.nextLine();

        System.out.println("Enter account number: ");
        int accountNumber = scanner.nextInt();

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

        BankAccount account = new BankAccount(name, accountNumber, balance);

        System.out.println("Account holder's name: " + account.getName());
        System.out.println("Account number: " + account.getAccountNumber());
        System.out.println("Balance: " + account.getBalance());

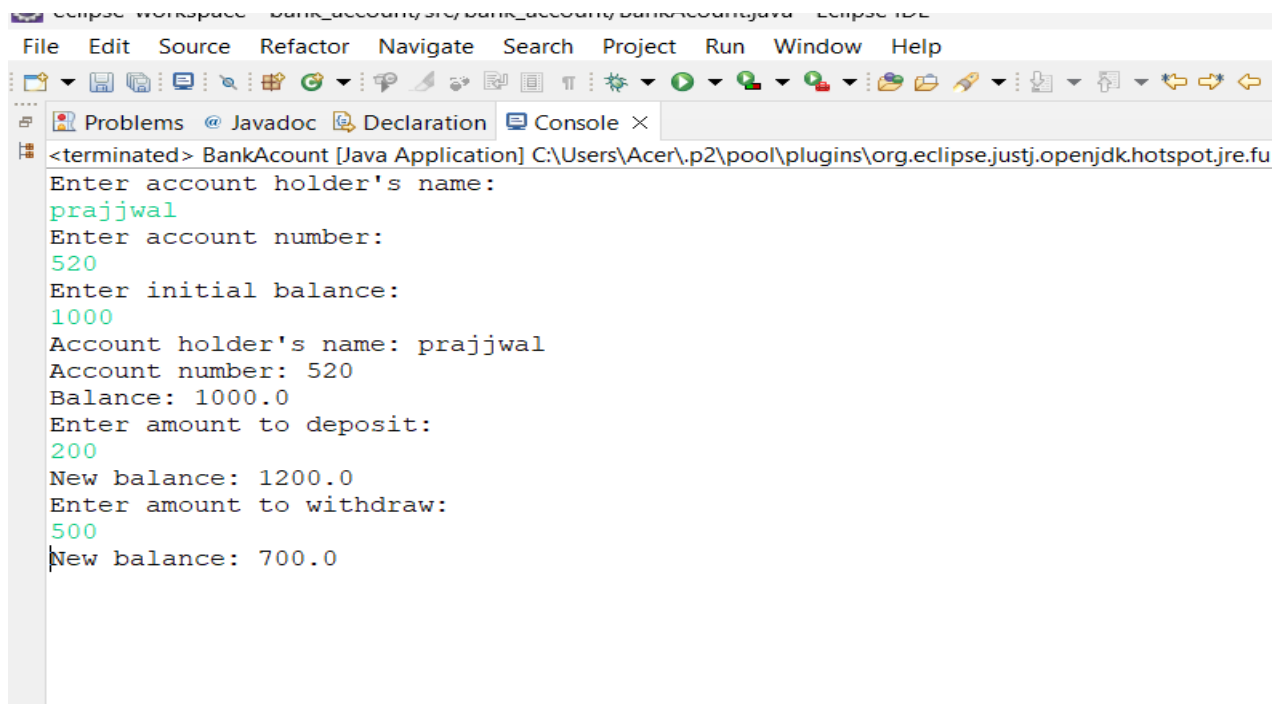
        System.out.println("Enter amount to deposit: ");
        double amountToDeposit = scanner.nextDouble();
        account.deposit(amountToDeposit);
        System.out.println("New balance: " + account.getBalance());

        System.out.println("Enter amount to withdraw: ");
        double amountToWithdraw = scanner.nextDouble();
```

```

        account.withdraw(amountToWithdraw);
        System.out.println("New balance: " + account.getBalance());
    }
}

```



```

Eclipse Workspace  BankAccount.java  Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
<terminated> BankAccount [Java Application] C:\Users\Acer\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.fu
Enter account holder's name:
prajjwal
Enter account number:
520
Enter initial balance:
1000
Account holder's name: prajjwal
Account number: 520
Balance: 1000.0
Enter amount to deposit:
200
New balance: 1200.0
Enter amount to withdraw:
500
New balance: 700.0

```

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

```

package override;

class Animal{
    public void speakSound() {
        System.out.println("Sound of animal");
    }
}

class Dog extends Animal{
    @Override
    public void speakSound() {
        System.out.println("Woof!!");
    }
}

class Cat extends Animal{
    @Override
    public void speakSound() {
        System.out.println("Mou!!");
    }
}

class Goat extends Animal{
    @Override
    public void speakSound() {
        System.out.println("Me!!");
    }
}

public class Main {
    public static void main(String[] args) {
        Animal aa = new Animal();
        aa.speakSound();

        Dog dd = new Dog();
        dd.speakSound();
    }
}

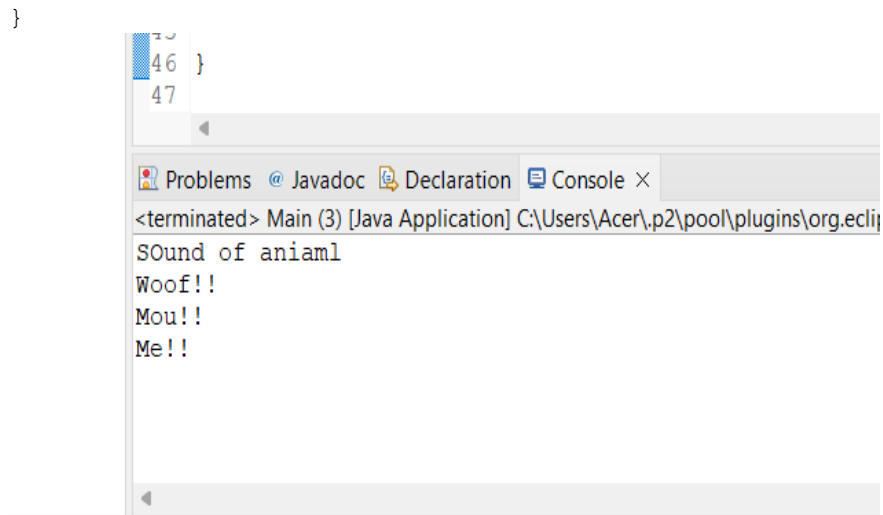
```

```

        Cat cc = new Cat();
        cc.speakSound();

        Goat gg = new Goat();
        gg.speakSound();
    }
}

```



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

```

package override;

class Animal {

    public void speak() {
        System.out.println("I am an animal");
    }
}

class Dog extends Animal {

    @Override
    public void speak() {
        System.out.println("Woof!");
    }
}

class Cat extends Animal {

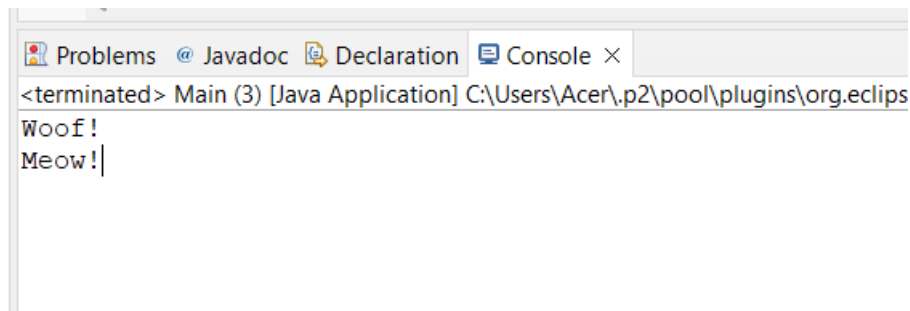
    @Override
    public void speak() {
        System.out.println("Meow!");
    }
}

public class Main {

    public static void main(String[] args) {
        Animal animal = new Dog();
        animal.speak();

        animal = new Cat();
        animal.speak();
    }
}

```



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

```
package override;

class Animal {

    public void speak() {
        System.out.println("I am an animal");
    }
    public void eat() {
        System.out.println("I am eating");
    }
}

class Dog extends Animal {

    @Override
    public void speak() {
        System.out.println("Woof!");
    }
    @Override
    public void eat() {
        System.out.println("Dog eating");
    }
}

class Cat extends Animal {

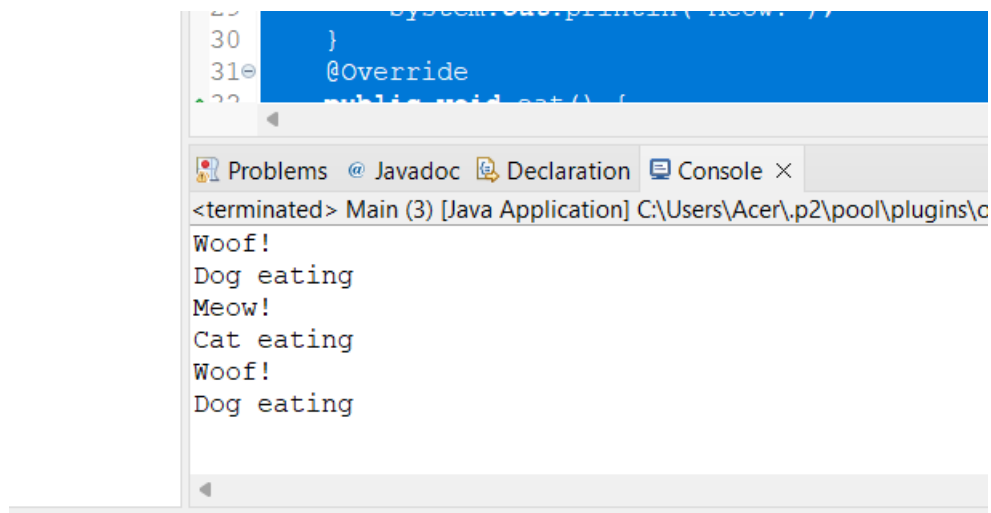
    @Override
    public void speak() {
        System.out.println("Meow!");
    }
    @Override
    public void eat() {
        System.out.println("Cat eating");
    }
}

public class Main {

    public static void main(String[] args) {
        Animal animal = new Dog();
        animal.speak();
        animal.eat();

        animal = new Cat();
        animal.speak();
        animal.eat();

        animal = new Dog();
        animal.speak();
        animal.eat();
    }
}
```



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

```
package loosecoupling;

interface Shape {

    void draw();
}

class Circle implements Shape {

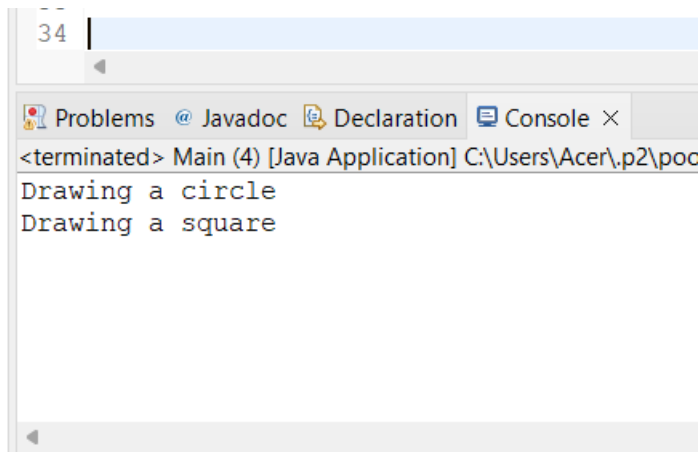
    @Override
    public void draw() {
        System.out.println("Drawing a circle");
    }
}

class Square implements Shape {

    @Override
    public void draw() {
        System.out.println("Drawing a square");
    }
}

public class Main {

    public static void main(String[] args) {
        Shape shape = new Circle();
        shape.draw();
        shape = new Square();
        shape.draw();
    }
}
```



Q6. What is the benefit of encapsulation in java?

- 1 Data hiding
2. Improved Security
3. Improved Readability
4. Improved Flexibility

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

- **Primitive data types:** Java supports **primitive data types**, such as **int, double, and char**. These data types are **not objects**, and **they do not inherit from any class**. This means that **they cannot be** used in some object-oriented programming concepts, such as **polymorphism and inheritance**.
- **Static methods:** Static methods are methods **that are associated with a class**, but they are not associated with any **particular object** of the class. Static methods cannot access the **instance variables of objects**, and they cannot be **overridden by subclasses**. This means that static methods are not fully object-oriented.
- **Final classes:** Final classes cannot be **subclassed**. This means that final classes cannot be used to implement the principle of polymorphism.

Q8.What are the advantages of abstraction in java?

- 1.Reduce the complexity
- 2.Increase reusability
- 3.Improved flexibility
- 4.Improved readability

Q9.What is an abstraction explained with an Example?

Abstraction: It is a object oriented programming language's concept use to hide the implementation of details of the class. It increase the readability ,flexibility maintainability of code.

```
abstract class Shape {
```

```

    abstract void draw();
}

class Circle extends Shape {

    @Override
    public void draw() {
        System.out.println("Drawing a circle");
    }
}

class Square extends Shape {

    @Override
    public void draw() {
        System.out.println("Drawing a square");
    }
}

public class Main {

    public static void main(String[] args) {
        Shape shape = new Circle();
        shape.draw();

        shape = new Square();
        shape.draw();
    }
}

```

Q10.What is the final class in Java?

Ans: Final class is , which define by using the “final” keyword and final class cannot be subclass class. It does not implement in polymorphism

Syntax: final class Name{

 //code block

}