Core Specific- Java Assignment 3

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

Ans:

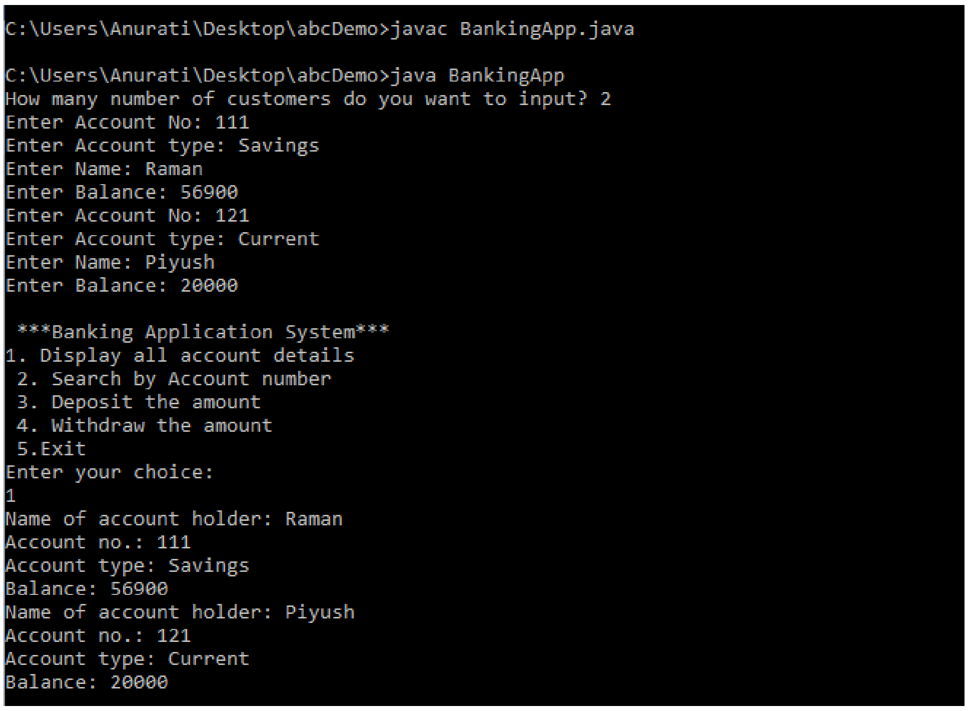
BankingApp.java

```
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.. ! !");
```

```java
                    }
                    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);
    }
}
```

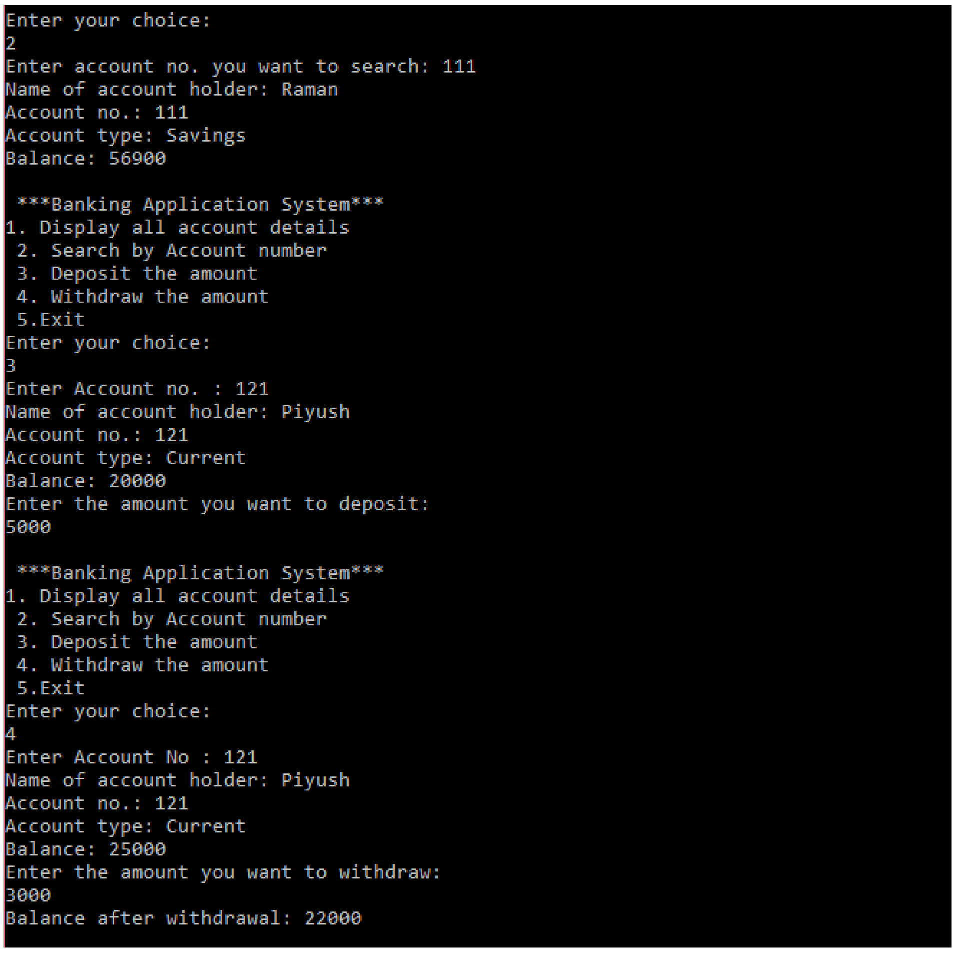
Core Specific- Java Assignment 3

Output 1:



```
C:\Users\Anurati\Desktop\abcDemo>javac BankingApp.java

C:\Users\Anurati\Desktop\abcDemo>java BankingApp
How many number of customers do you want to input? 2
Enter Account No: 111
Enter Account type: Savings
Enter Name: Raman
Enter Balance: 56900
Enter Account No: 121
Enter Account type: Current
Enter Name: Piyush
Enter Balance: 20000

 ***Banking Application System***
1. Display all account details
 2. Search by Account number
 3. Deposit the amount
 4. Withdraw the amount
 5.Exit
Enter your choice:
1
Name of account holder: Raman
Account no.: 111
Account type: Savings
Balance: 56900
Name of account holder: Piyush
Account no.: 121
Account type: Current
Balance: 20000
```

Output 2:



```
Enter your choice:
2
Enter account no. you want to search: 111
Name of account holder: Raman
Account no.: 111
Account type: Savings
Balance: 56900

 ***Banking Application System***
1. Display all account details
 2. Search by Account number
 3. Deposit the amount
 4. Withdraw the amount
 5.Exit
Enter your choice:
3
Enter Account no. : 121
Name of account holder: Piyush
Account no.: 121
Account type: Current
Balance: 20000
Enter the amount you want to deposit:
5000

 ***Banking Application System***
1. Display all account details
 2. Search by Account number
 3. Deposit the amount
 4. Withdraw the amount
 5.Exit
Enter your choice:
4
Enter Account No : 121
Name of account holder: Piyush
Account no.: 121
Account type: Current
Balance: 25000
Enter the amount you want to withdraw:
3000
Balance after withdrawal: 22000
```

Core Specific- Java Assignment 3

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

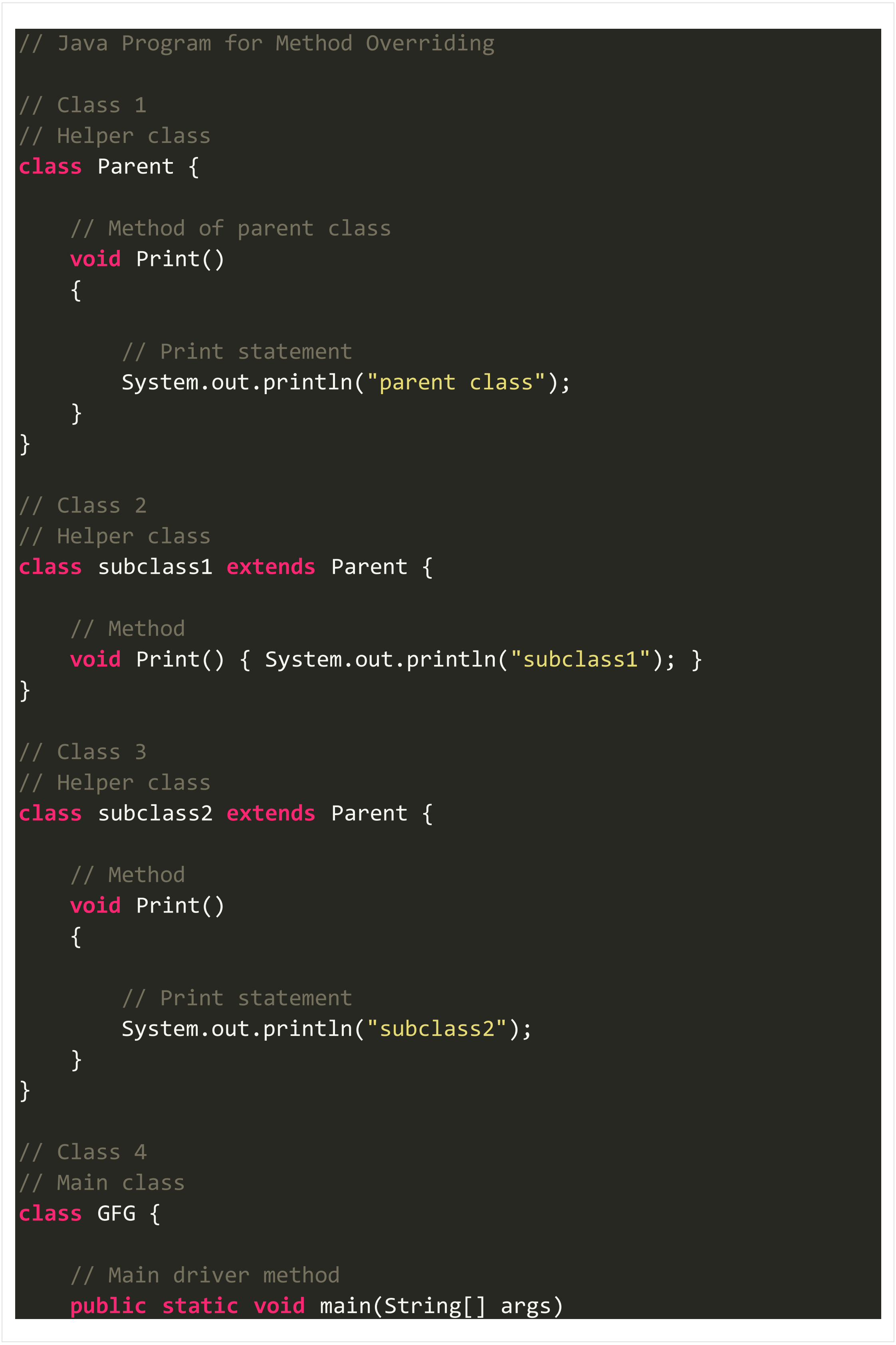
Ans:

```java
// A Simple Java program to demonstrate
// method overriding in java

// Base Class
class Parent {
    void show() { System.out.println("Parent's show()"); }
}

// Inherited class
class Child extends Parent {
    // This method overrides show() of Parent
    @Override void show()
    {
        System.out.println("Child's show()");
    }
}

// Driver class
class Main {
    public static void main(String[] args)
    {
        // If a Parent type reference refers
        // to a Parent object, then Parent's
        // show is called
        Parent obj1 = new Parent();
        obj1.show();

        // If a Parent type reference refers
        // to a Child object Child's show()
        // is called. This is called RUN TIME
        // POLYMORPHISM.
        Parent obj2 = new Child();
        obj2.show();
    }
}
```

Core Specific- Java Assignment 3

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

Ans:



```java
// Java Program for Method Overriding

// Class 1
// Helper class
class Parent {

    // Method of parent class
    void Print()
    {

        // Print statement
        System.out.println("parent class");
    }
}

// Class 2
// Helper class
class subclass1 extends Parent {

    // Method
    void Print() { System.out.println("subclass1"); }
}

// Class 3
// Helper class
class subclass2 extends Parent {

    // Method
    void Print()
    {

        // Print statement
        System.out.println("subclass2");
    }
}

// Class 4
// Main class
class GFG {

    // Main driver method
    public static void main(String[] args)
    {

        // Creating object of class 1
        Parent a;

        // Now we will be calling print methods
        // inside main() method

        a = new subclass1();
        a.Print();

        a = new subclass2();
        a.Print();
    }
}
```

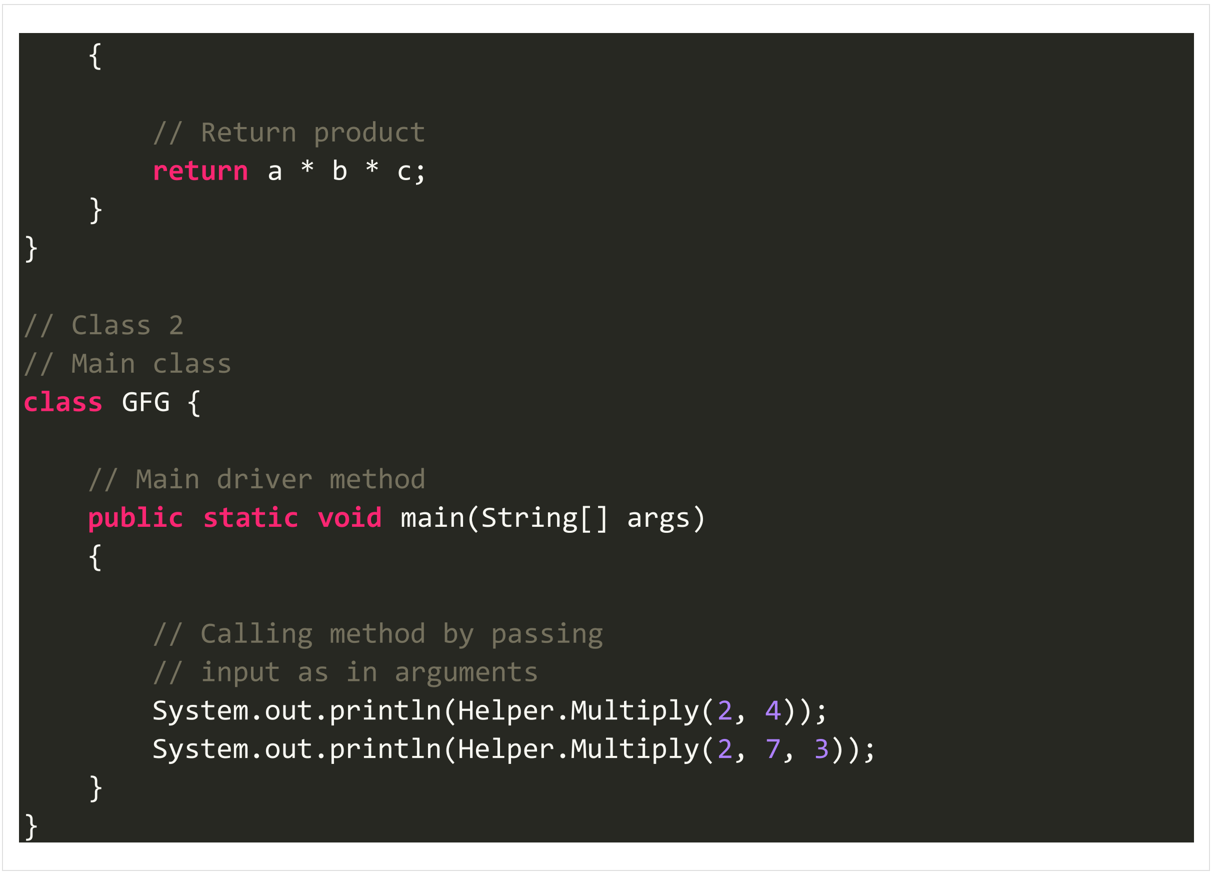
Output

```
subclass1

subclass2
```

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

Ans:

```
// Java program for Method Overloading
// by Using Different Numbers of Arguments

// Class 1
// Helper class
class Helper {

    // Method 1
    // Multiplication of 2 numbers
    static int Multiply(int a, int b)
    {

        // Return product
        return a * b;
    }

    // Method 2
    // // Multiplication of 3 numbers
    static int Multiply(int a, int b, int c)
```



```
    {

        // Return product
        return a * b * c;
    }
}

// Class 2
// Main class
class GFG {

    // Main driver method
    public static void main(String[] args)
    {

        // Calling method by passing
        // input as in arguments
        System.out.println(Helper.Multiply(2, 4));
        System.out.println(Helper.Multiply(2, 7, 3));
    }
}
```

Output

```
8

42
```

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

Ans:

```
// parent or base class
class A
{
void foo()
{
  System.out.println("Inside the foo method of base class.");
}
}

// child or derived class
class B extends A
```

```
{
void foo()
{
  System.out.println("Inside the foo method of derived class.");
}
}

public class CouplingExample
{
// main method
public static void main(String argvs[])
{

// creating an object of class B
B obj = new B();
obj.foo();
}
}
```

Output:

```
Inside the foo method of derived class.
```

Q6. What is the benefit of encapsulation in java?

Ans:

Encapsulation prevents access to data members and data methods by any external classes. The encapsulation process improves the security of the encapsulated data.

Core Specific- Java Assignment 3

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.). There are seven 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 predefined types are objects
6. All user defined types are objects
7. All operations performed on objects must be only through methods exposed at the objects.

Why Java is not a Pure Object-Oriented Language?

Java supports property 1, 2, 3, 4 and 6 but fails to support property 5 and 7 given above.

Java language is not a Pure Object-Oriented Language as it contains these properties.

Q8. What are the advantages of abstraction in java?

Ans:

Advantages of Abstract Classes

- Abstract class in Java is highly beneficial in writing shorter codes.
- Abstraction in Java avoids code duplication.
- Abstract classes enable code reusability.
- Changes to internal code implementation are done without affecting classes.

Core Specific- Java Assignment 3

Q9. What is an abstraction explained with an Example?

Ans:

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery.

Abstraction lets you focus on what the object does instead of how it does it.

Ways to achieve Abstraction

There are two ways to achieve abstraction in java

1. Abstract class (0 to 100%)
2. Interface (100%)

Example:

```java
abstract class Bike{
abstract void run();
}
class Honda4 extends Bike{
void run(){System.out.println("running safely");}
public static void main(String args[]){
Bike obj = new Honda4();
obj.run();
}
}
```

Output:

```
running safely
```

Core Specific- Java Assignment 3

Q10. What is the final class in Java?

Ans:

The final class is a class that is declared with the final keyword. We can restrict class inheritance by making use of the final class. Final classes cannot be extended or inherited. If we try to inherit a final class, then the compiler throws an error during compilation.