1. **Class, Objects, Variables, Methods & Constructors Coding Questions**

**Que 01) Java Program to Illustrate Use of All Features of Abstract Class**

**Ans🡪**

//An abstract class can have a data member,

//abstract method, method body (non-abstract method),

// constructor, and even main() method.

abstract class Bike {

Bike() {

System.out.println("bike is created");

}

abstract void run();

void changeGear() {

System.out.println("gear changed");

}

}

class Honda extends Bike {

void run() {

System.out.println("running safely...");

}

}

class Abstraction {

public static void main(String args[]) {

Bike obj=new Honda();

obj.run();

obj.changeGear();

}

}

/\*E:\CDAC\Interview>java Abstraction

bike is created

running safely...

gear changed

E:\CDAC\Interview>

\*/

**Que 02) Java Program to Illustrates Use of Abstract Class and Method**

**Ans🡪**

abstract class Bank {

abstract int getRateOfInterest();

}

class SBI extends Bank {

int getRateOfInterest() {

return 7;

}

}

class PNB extends Bank {

int getRateOfInterest() {

return 8;

}

}

class Abstraction2 {

public static void main(String args[]) {

Bank b;

b=new SBI();

System.out.println("Rate of Interest is: "+b.getRateOfInterest()+" %");

b=new PNB();

System.out.println("Rate of Interest is: "+b.getRateOfInterest()+" %");

}

}

/\*

E:\CDAC\Interview>java Abstraction2

Rate of Interest is: 7 %

Rate of Interest is: 8 %

E:\CDAC\Interview>

\*/

**Que 03) Java Program to Illustrates Use of Instance Inner Class**

**Ans🡪**

class Batsman

{

int totRuns;

int totMatches;

Batsman()

{

}

Batsman(int totRuns, int totMatches)

{

this.totRuns = totRuns;

this.totMatches = totMatches;

}

void printAverage()

{

AvgCalculator avgcal = new AvgCalculator();

double res = avgcal.calAverage();

System.out.println("Batsman Average is : "+res);

}

class AvgCalculator

{

double avg;

double calAverage()

{

avg = totRuns/totMatches;

return avg;

}

}

}

class InnerClassDemo

{

public static void main (String args [])

{

Batsman bats1 = new Batsman(1500,10);

bats1.printAverage();

}

}

**Que 04) Java Program to Illustrates Use of Static Inner Class**

**Ans🡪**

class Batsman

{

static int totRuns;

int totMatches;

Batsman()

{

}

Batsman(int totRuns, int totMatches)

{

this.totRuns = totRuns;

this.totMatches = totMatches;

}

void printAverage()

{

AvgCalculator avgcal = new AvgCalculator();

avgcal.myFun();

}

static class AvgCalculator

{

void myFun()

{

System.out.println("My Fun Of AverageCalculator");

int res = totRuns + 100;

System.out.println("total runs is : "+res);

}

}

}

class InnerClassDemo2

{

public static void main (String args [])

{

Batsman bats1 = new Batsman(1500,10);

bats1.printAverage();

}

}

**Que 05) Java Program to Illustrates Use of Referencing the Object from Inner Class**

**Ans🡪**

import java.lang.\*;

import java.util.\*;

class Car{

static String manufacturer;

static String color;

static int avg;

static int noOfDoors;

Car()

{

}

Car(String manufacturer, String color, int avg, int noOfDoors){

this.manufacturer = manufacturer;

this.color = color;

this.avg = avg;

this.noOfDoors = noOfDoors;

}

static void printDetails()

{

System.out.println("Name of the manufacturer of the car is : "+manufacturer);

System.out.println("Color of the car is : "+color);

System.out.println("Average of the car is : "+avg);

System.out.println("No of doors in the car are : "+noOfDoors);

}

class CarDetails

{

void methodCall()

{

printDetails();

}

}

}

class ReferencingObject5{

public static void main(String args[]){

Car c1 = new Car("TATA", "Red", 15, 4);

Car.CarDetails ccd1 = c1.new CarDetails();

ccd1.methodCall();

}

}

**Que 06) Java Program to Create Outer Class Bank Account and the Inner Class Interest in it**

**Ans🡪**

import java.lang.\*;

import java.util.\*;

class BankAccount{

String nameOfAccountHolder;

long accountNo;

double balance;

int interestRate;

BankAccount(){

}

BankAccount(String nameOfAccountHolder, long accountNo, double balance, int interestRate){

this.nameOfAccountHolder = nameOfAccountHolder;

this.accountNo = accountNo;

this.balance = balance;

this.interestRate = interestRate;

}

void printAccountDetails(double totInterest){

System.out.println("Name of the account holder is : "+nameOfAccountHolder);

System.out.println("Account no is : "+accountNo);

System.out.println("Balance in the account is : "+balance);

System.out.println("Total interest per year is : "+totInterest);

}

class Interest{

double calInterest(){

double totInterest = ( balance \* interestRate )/ 100;

return totInterest;

}

}

}

class BankAccountOuter6{

public static void main(String args[]){

BankAccount ba1 = new BankAccount("Omkar", 123456, 20000, 7);

BankAccount.Interest bai1 = ba1.new Interest();

double res = bai1.calInterest();

ba1.printAccountDetails(res);

}

}

**Que 07) Java Program to Implement Shape Interface using Circle and Rectangle Class**

**Ans**🡪

import java.lang.\*;

import java.util.\*;

interface Shape{

void printDetails();

void calArea();

}

class Rectangle implements Shape{

int length;

int breadth;

Rectangle(){

}

Rectangle(int length, int breadth){

this.length = length;

this.breadth = breadth;

}

public void printDetails(){

System.out.println("Length of the rectangle is : "+length);

System.out.println("Breadth of the rectangle is : "+breadth);

}

public void calArea(){

int areaR = length \* breadth;

System.out.println("Area of the rectangle is : "+areaR);

}

}

class Circle implements Shape{

double radius;

Circle(){

}

Circle(double radius){

this.radius = radius;

}

public void printDetails(){

System.out.println("Radius of the circle is : "+radius);

}

public void calArea(){

double areaC = (double)2\*3.14\*radius;

System.out.println("Area of the circle is : "+areaC);

}

}

class ShapeInterface7{

public static void main(String args[]){

Rectangle r1 = new Rectangle(4, 5);

r1.printDetails();

r1.calArea();

Circle c1 = new Circle(4.6);

c1.printDetails();

c1.calArea();

}

}

**Que 08) Java Program to Count Number of Objects Created for Class**

**Ans**🡪

import java.lang.\*;

import java.util.\*;

class Student{

static int count;

Student(){

count++;

}

static void printCount(){

System.out.println("No of objects created for the class is : "+count);

}

}

class ClassObjectCount8{

public static void main(String args[]){

Student s1 = new Student();

Student s2 = new Student();

Student s3 = new Student();

Student s4 = new Student();

Student s5 = new Student();

Student s6 = new Student();

Student.printCount();

}

}

**Que 09) Java Program to Implement the Passing and Returning Objects**

**Ans🡪**

-->Java Program to Demonstrate Objects Passing to Methods.

class ObjectPassDemo {

int a, b;

// Constructor

ObjectPassDemo(int i, int j)

{

a = i;

b = j;

}

// Method

boolean equalTo(ObjectPassDemo o)

{

// Returns true if o is equal to the invoking

// object notice an object is passed as an

// argument to method

return (o.a == a && o.b == b);

}

}

// Main class

public class Que09 {

// MAin driver method

public static void main(String args[])

{

// Creating object of above class inside main()

ObjectPassDemo ob1 = new ObjectPassDemo(100, 22);

ObjectPassDemo ob2 = new ObjectPassDemo(100, 22);

ObjectPassDemo ob3 = new ObjectPassDemo(-1, -1);

// Checking whether object are equal as custom

// values

// above passed and printing corresponding boolean

// value

System.out.println("ob1 == ob2: "

+ ob1.equalTo(ob2));

System.out.println("ob1 == ob3: "

+ ob1.equalTo(ob3));

}

}

/\*

**OUTPUT -**

F:\CDAC\CDAC\_Lab\Diwali Homework\Pattern Assignment\Class, Objects, Variables, Methods & Constructors Coding Questions>java Passing\_and\_Returning\_Objects

ob1 == ob2: true

ob1 == ob3: false

F:\CDAC\CDAC\_Lab\Diwali Homework\Pattern Assignment\Class, Objects, Variables, Methods & Constructors Coding Questions>\*/

**Que10. Java Program to Swap Objects using Swap() Method**

**Ans**🡪

/\*

--> Java program to demonstrate that we can swap two objects be swapping members

-->Direct Object Swapping - is not swaps all parameters present in any object

\*/

// objects be swapping members

// Where it does not work

// Class 1

// A car with number and name

class Car {

// Attributes of Car class

int no;

String model;

// Constructor

Car()

{

this.model = "No Car";

this.no= 0;

}

Car(String model, int no)

{

// This keyword is used to refer

// current instance itself

this.model = model;

this.no = no;

}

}

// Class 2

// A class that uses Car

class Que10

{

// swap() doesn't swap c1 and c2

public static void swap(Car c1, Car c2)

{

Car temp = c1;

c1 = c2;

c2 = temp;

System.out.println("in swap() method call-");

System.out.println("Obj1 : no = " + c1.no +", model = " + c1.model);

System.out.println("Obj2 : no = " + c2.no +", model = " + c2.model);

}

// Driver method

public static void main(String[] args)

{

Car c1 = new Car("Audi", 1);

Car c2 = new Car("BMW", 2);

System.out.println("Current Objects-");

System.out.println("Obj1 : no = " + c1.no +", model = " + c1.model);

System.out.println("Obj2 : no = " + c2.no +", model = " + c2.model);

// c1.print();

// c2.print();

swap(c1, c2);

System.out.println("After Swapping the Object in main method-");

System.out.println("Obj1 : no = " + c1.no +", model = " + c1.model);

System.out.println("Obj2 : no = " + c2.no +", model = " + c2.model);

// c1.print();

// c2.print();

}

}

/\*

**OUTPUT-** object are not swapped by direct swapping of object using Swap()method.

F:\CDAC\CDAC\_Lab\Diwali Homework\Pattern Assignment\Class, Objects, Variables>java Que10

Current Objects-

Obj1 : no = 1, model = Audi

Obj2 : no = 2, model = BMW

in swap() method call-

Obj1 : no = 2, model = BMW

Obj2 : no = 1, model = Audi

After Swapping the Object in main method-

Obj1 : no = 1, model = Audi

Obj2 : no = 2, model = BMW

F:\CDAC\CDAC\_Lab\Diwali Homework\Pattern Assignment\Class, Objects, Variables>

F:\CDAC\CDAC\_Lab\Diwali Homework\Pattern Assignment\Class, Objects, Variables>

\*/

**Ans🡪A--> Using concepts of OOPS**

\*/

// Java program to demonstrate that we can swap two

// objects be swapping members

class Car {

// Attributes associated with car

int no;

Car(int no) { this.no = no; }

}

// Uses Car objects

class Que10A {

// Method 1

// To swap

public static void swap(Car c1, Car c2)

{

int temp = c1.no;

c1.no = c2.no;

c2.no = temp;

}

// Method 2

// Main driver method

public static void main(String[] args)

{

// Creating car class objects(creating cars)

Car c1 = new Car(1);

Car c2 = new Car(2);

System.out.println("Current Objects-");

System.out.println("c1.no = " + c1.no);

System.out.println("c2.no = " + c2.no);

// Calling method 1

swap(c1, c2);

System.out.println("After Swapping the Object-");

// Print and display commands

System.out.println("c1.no = " + c1.no);

System.out.println("c2.no = " + c2.no);

}

}

/\*

**OUTPUT-** by Swaping objects actual parameters ae swapped only with eachparameter swapping and not whole object.

F:\CDAC\CDAC\_Lab\Diwali Homework\Pattern Assignment\Class, Objects, Variables>java Que10A

Current Objects-

c1.no = 1

c2.no = 2

After Swapping the Object-

c1.no = 2

c2.no = 1

**OUTPUT 2-**

by Swaping objects actual parameters ae not swapped. to overcome this we need Wrapper class of object.

F:\CDAC\CDAC\_Lab\Nov 03 2022>java Que10

Current Objects-

c1.no = 1

c2.no = 2

After Swapping the Object-

c1.no = 2

c2.no = 1

\*/

**Ans🡪B--> Can be Used to Swap two Objects**

class Car {

// Attributes associated with car

int no;

String model;

// Constructor of class 1

Car(String model, int no)

{

// This refers to current instance itself

this.model = model;

this.no = no;

}

}

// Class 2

// Wrapper over class that is used for swapping

class CarWrapper {

Car c;

// Constructor

CarWrapper(Car c) { this.c = c; }

}

// Class 3

// Uses Car class and swaps objects of Car using CarWrapper

class Que10B {

// This method swaps car objects in wrappers cw1 and cw2

public static void swap(CarWrapper cw1, CarWrapper cw2)

{

Car temp = cw1.c;

cw1.c = cw2.c;

cw2.c = temp;

}

// Main driver method

public static void main(String[] args)

{

Car c1 = new Car("Ferrari", 3);

Car c2 = new Car("Bugati", 4);

//C1 and C2 assigned in wrapper class

CarWrapper cw1 = new CarWrapper(c1);

CarWrapper cw2 = new CarWrapper(c2);

System.out.println("Before Swapping the Object in main method-");

System.out.println("Obj1 : no = " + c1.no +", model = " + c1.model);

System.out.println("Obj2 : no = " + c2.no +", model = " + c2.model);

swap(cw1, cw2); //swapping of 2 object of wrapper class.

// assignback wrapper class object to original objects.

c1=cw1.c;

c2=cw2.c;

// printing original object after swapping

System.out.println("After Swapping the Object in main method-");

System.out.println("Obj1 : no = " + c1.no +", model = " + c1.model);

System.out.println("Obj2 : no = " + c2.no +", model = " + c2.model);

}

}

/\*

**OUTPUT -**

F:\CDAC\CDAC\_Lab\Diwali Homework\Pattern Assignment\Class, Objects, Variables>java Que10B

**Before Swapping the Object in main method-**

Obj1 : no = 3, model = Ferrari

Obj2 : no = 4, model = Bugati

**After Swapping the Object in main method-**

Obj1 : no = 4, model = Bugati

Obj2 : no = 3, model = Ferrari

\*/

**Que 11) Java Program to Illustrate Use of Methods in a Class**

**Ans**🡪

import java.lang.\*;

import java.util.\*;

class Demo{

int a;

int b;

Demo(){

}

Demo(int a, int b){

this.a = a;

this.b = b;

}

public void add(){

int res = a + b;

System.out.println("Addition of a and b is : "+res);

}

void sub(){

int res = a - b;

System.out.println("Subtraction of the a and b is : "+res);

}

public int mul(){

int res = a\*b;

return res;

}

}

class UseOfMethods11{

public static void main(String args[]){

Demo d1 = new Demo(10, 5);

d1.add();

d1.sub();

int i = d1.mul();

System.out.println("Multiplication of two number is : "+i);

}

}

**Que 12) Java Program to Create a Method without Parameters and with Return Type**

**Ans🡪**

import java.lang.\*;

import java.util.\*;

class Operations{

int a;

int b;

String name;

Operations(){

}

Operations(int a, int b, String name){

this.a = a;

this.b = b;

this.name = name;

}

public int addition(){

int res = a + b;

return res;

}

public double multiplication(){

double res = (double)a \* b;

return res;

}

public String printName(){

return name;

}

}

class MethodWithoutParaWithReturn12{

public static void main(String args[]){

Operations o1 = new Operations(4, 5, "Omkar");

int x = o1.addition();

double y = o1.multiplication();

String z = o1.printName();

System.out.println("Addition of two no is : "+x);

System.out.println("Multiplication of two no is : "+y);

System.out.println("Name is : "+z);

}

}

**Que 13) Java Program to Create a Method without Parameters and Return Type**

**Ans🡪**

class Main

{

void display()

{

System.out.println("Happy Diwali all of you.");

}

public static void main(String args[])

{

Main obj = new Main();

obj.display();

}

}

**Que 14) Java Program to Create a Method with 2 Parameters and without Return Type**

**Ans**🡪

class Main

{

void display(int a, int b)

{

System.out.println("Sum of a and b = "+(a+b));

}

public static void main(String args[])

{

Main obj = new Main();

obj.display(5,4);

}

}

**Que 15) Java Program to Illustrate the Use of HashCode() Method**

**Ans**🡪

public class Hash15

{

public static void main(String[] args)

{

Integer i= new Integer("255");

int hashValue =i.hashCode();

System.out.println("Hash code Value for object is:"+ hashValue);

}

}

**Que 16) Java Program to Illustrate Use of Final Keyword**

**Ans**🡪

import java.util.Scanner;

class Fin16

{

final int length = 6;

final int breadth = 4;

final void area()

{

int a = length\*breadth;

System.out.println("Area"+a);

}

}

class Rectangle extends Fin16

{

final void rect()

{

System.out.println("This is rectangle");

}

}

final public class Final\_Use extends Rectangle

{

public static void main(String[] args)

{

Final\_Use obj = new Final\_Use();

obj.rect();

obj.area();

}

}

**Que 17)** **Java Program to Illustrate Use of Constructor**

**Ans**🡪

import java.lang.\*;

import java.util.\*;

class Vehicles{

int noOfWheels;

String color;

Vehicles(){

}

Vehicles(int noOfWheels, String color){

this.noOfWheels = noOfWheels;

this.color = color;

}

public void printDetails(){

System.out.println("No of wheels in the vehicle are : "+noOfWheels);

System.out.println("Color of the vehicle is : "+color);

}

}

class Car extends Vehicles{

int maxSpeed;

Car(){

}

Car(int maxSpeed){

this.maxSpeed = maxSpeed;

}

public void carDetails(){

System.out.print("Max speed of the car is : "+maxSpeed);

}

}

class Constructors17{

public static void main(String args[]){

Vehicles v1 = new Vehicles(4, "Blue");

v1.printDetails();

Car c1 = new Car(120);

c1.carDetails();

}

}

**Que 18) Java Program to Illustrates Use of Chaining Constructor**

**Ans**🡪

import java.lang.\*;

import java.util.\*;

class Addition{

Addition(){

}

Addition(int a){

System.out.println("Value of the a is : "+a);

}

Addition(int a, int b){

this(a);

System.out.println("Addition of the a and b is : "+(a+b));

}

Addition(int a, int b, int c){

this(a, b);

System.out.println("Addition of the a, b and c is : "+(a+b+c));

}

}

class ConstructorChaining18{

public static void main(String args[]){

Addition a1 = new Addition(10, 20, 30);

}

}

**Que 19)** **Java Program to Create an Object for Class and Assign Value in the Object using Constructor**

**Ans**🡪

import java.lang.\*;

import java.util.\*;

class Student{

int rollNo;

String name;

double marks;

Student(){

}

Student(int rollNo, String name, double marks){

this.rollNo = rollNo;

this.name = name;

this.marks = marks;

}

public void printDetails(){

System.out.println("Roll no of the student is : "+rollNo);

System.out.println("Name of the student is : "+name);

System.out.println("Marks of the student is : "+marks);

}

}

class ObjectUsingConstructor19{

public static void main(String args[]){

Student s1 = new Student(1, "Omkar", 75.8);

s1.printDetails();

}

}

**Que 20)** **String Constructor Program in Java**

**Ans🡪**

import java.lang.\*;

import java.util.\*;

class Employee{

String type;

Employee(){

}

Employee(String type){

this.type = type;

}

void printDetails(){

System.out.println("Given string is : "+type);

}

}

class StringConstructor20{

public static void main(String args[]){

Employee e1 = new Employee("Omkar");

e1.printDetails();

}

}

**Que 21) Java Program to Allocate and Initialize Super Class Members using Constructor**

**Ans**🡪

import java.lang.\*;

import java.util.\*;

class First{

int a;

int b;

First(){

}

First(int a, int b){

this.a = a;

this.b = b;

}

}

class Second extends First{

int c;

Second(){

}

Second(int a, int b, int c){

super(a, b);

this.c = c;

}

void printValues(){

System.out.println("Value of a is : "+a);

System.out.println("Value of b is : "+b);

System.out.println("Value of c is : "+c);

}

}

class CallingSuperClassConstructor21{

public static void main(String args[]){

Second s1 = new Second(5, 6, 7);

s1.printValues();

}

}

**Que 22) Java Program to Check the Accessibility of Static and Non-Static Variable by a Static Method**

**Ans**🡪

class Demo

{

int a=10;

static int b=20;

public static void main(String args[])

{

//System.out.println("a= "+a); //error: non-static variable a cannot be referenced from a static context

System.out.println("b= "+b);

Demo obj = new Demo();

System.out.println("a= "+obj.a);

}

}

//From static method, we have to create object and from that object we can access the non-static variable.

**Que 23) Java Program to Demonstrate Usage of an Instance Variable in the Test Class**

**Ans**🡪

class Instance

{

int x=17;

public static void main(String...a)

{

Instance ob = new ();

System.out.println(ob.x);

}

}

**Que 24) Java Program to Demonstrate Usage of a Static Variable in the Test Class**

**Ans**🡪

import java.lang.\*;

import java.util.\*;

class Test{

static String name;

static String cityName;

static String stateName;

Test(){

}

Test(String name, String cityName, String stateName){

this.name = name;

this.cityName = cityName;

this.stateName = stateName;

}

static void printData(){

System.out.println("Your name is : "+name);

System.out.println("You live in : "+cityName);

System.out.println("You are from : "+stateName);

}

}

class StaticVariables24{

public static void main(String args[]){

Test t1 = new Test("Omkar", "Mumbai", "Maharashtra");

t1.printData();

}

}

**Que 25) Java Program to Check Whether Which One is Executed First, Static Block or the Static Method**

**Ans**🡪

import java.lang.\*;

import java.util.\*;

class First{

static int a;

static int b;

static{

System.out.println("Static block - 1");

}

static void staticMethod(){

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

}

static{

System.out.println("Static block - 2");

}

}

class StaticBlockStaticMethod25{

public static void main(String args[]){

First.staticMethod();

}

}

**Que26) Java Program to Calculate Sum of Two Byte Values using Type Casting**

**Ans**🡪

import java.lang.\*;

import java.util.\*;

class AdditonOfByte26{

public static void main(String args[]){

Scanner sc = new Scanner(System.in);

System.out.println("Enter the first byte number : (-128 to 127) ");

byte a = sc.nextByte();

System.out.println("Enter the second byte number : (-128 to 127) ");

byte b = sc.nextByte();

byte res = (byte)(a+b);

System.out.println("Sum of given byte numbers is : "+res);

}

}