1. Write a java program to implement scanner class

import java.util.Scanner;

class Main {

public static void main(String[] args) {

Scanner myObj = new Scanner(System.in);

System.out.println("Enter name, age and salary:");

// String input

String name = myObj.nextLine();

// Numerical input

int age = myObj.nextInt();

double salary = myObj.nextDouble();

// Output input by user

System.out.println("Name: " + name);

System.out.println("Age: " + age);

System.out.println("Salary: " + salary);

}

}

**Output:**

Enter name, age and salary:

Ram

32

30000

Name: Ram

Age: 32

Salary: 30000.0

2. Write a java program to perform addition of two numbers using arrays.

class Testarray5{

public static void main(String args[]){

//creating two matrices

int a[][]={{1,3,4},{3,4,5}};

int b[][]={{1,3,4},{3,4,5}};

int c[][]=new int[2][3];

for(int i=0;i<2;i++){

for(int j=0;j<3;j++){

c[i][j]=a[i][j]+b[i][j];

System.out.print(c[i][j]+" ");

}

System.out.println();//new line

}

}

}

**Output**

2 6 8

6 8 10

3. Write a java program to implement constructor overloading.

**class** Student5{

**int** id;

    String name;

**int** age;

    //creating two arg constructor

    Student5(**int** i,String n){

    id = i;

    name = n;

    }

    //creating three arg constructor

    Student5(**int** i,String n,**int** a){

    id = i;

    name = n;

    age=a;

    }

**void** display(){System.out.println(id+" "+name+" "+age);}

**public** **static** **void** main(String args[]){

    Student5 s1 = **new** Student5(111,"Karan");

    Student5 s2 = **new** Student5(222,"Aryan",25);

    s1.display();

    s2.display();

   }

}

**Output**

111 Karan 0

222 Aryan 25

4. Write a java program to implement static method.

class Student{

int rollno;

String name;

static String college = "ITS";

static void change(){

college = "BBDIT";

}

Student(int r, String n){

rollno = r;

name = n;

}

void display(){System.out.println(rollno+" "+name+" "+college);}

}

public class TestStaticMethod{

public static void main(String args[]){

Student.change();//calling change method

//creating objects

Student s1 = new Student(111,"Karan");

Student s2 = new Student(222,"Aryan");

Student s3 = new Student(333,"Sonoo");

//calling display method

s1.display();

s2.display();

s3.display();

}

}

**Output**

111 Karan BBDIT

222 Aryan BBDIT

333 Sonoo BBDIT

5. Write a java program to invoke current class constructor.

**class** A{

A(){System.out.println("hello a");}

A(**int** x){

**this**();

System.out.println(x);

}

}

**class** TestThis5{

**public** **static** **void** main(String args[]){

A a=**new** A(10);

}

}

**Output**

hello a

10

6. Write a java which implements Calling parameterized constructor from default constructor.

**class** A{

A(){

**this**(5);

System.out.println("hello a");

}

A(**int** x){

System.out.println(x);

}

}

**class** TestThis6{

**public** **static** **void** main(String args[]){

A a=**new** A();

}}

Output

hello a

7. Write a java which implements Calling parameterized constructor from default constructor.

**class** Student{

**int** rollno;

String name,course;

**float** fee;

Student(**int** rollno,String name,String course){

**this**.rollno=rollno;

**this**.name=name;

**this**.course=course;

}

Student(**int** rollno,String name,String course,**float** fee){

**this**.fee=fee;

**this**(rollno,name,course);//C.T.Error

}

**void** display(){System.out.println(rollno+" "+name+" "+course+" "+fee);}

}

**class** TestThis8{

**public** **static** **void** main(String args[]){

Student s1=**new** Student(111,"ankit","java");

Student s2=**new** Student(112,"sumit","java",6000f);

s1.display();

s2.display();

}

}

**Output**

TestThis8.java:12: error: call to this must be first statement in constructor

**this**(rollno,name,course);

8. Write a java program to implement Single Inheritance

class Animal{

void eat(){System.out.println("eating...");}

}

class Dog extends Animal{

void bark(){System.out.println("barking...");}

}

class TestInheritance{

public static void main(String args[]){

Dog d=new Dog();

d.bark();

d.eat();

}}

**Output**

barking...

eating...

9. Write a java program to implement Multi level Inheritance

class Animal{

void eat(){

System.out.println("eating...");

}

}

class Dog extends Animal{

void bark(){

System.out.println("barking...");

}

}

class BabyDog extends Dog{

void weep(){

System.out.println("weeping...");

}

}

class TestInheritance2{

public static void main(String args[]){

BabyDog d=new BabyDog();

d.weep();

d.bark();

d.eat();

}

}

**Output**

weeping...

barking...

eating...

10. Write a java program to implement Hierarical Inheritance

class Animal{

void eat(){

System.out.println("eating...");

}}

class Dog extends Animal{

void bark(){

System.out.println("barking...");

}}

class Cat extends Animal{

void meow(){

System.out.println("meowing...");

}}

class TestInheritance3{

public static void main(String args[]){

Cat c=new Cat();

c.meow();

Dog d=new Dog();

c.eat();

d.bark();

d.eat();

}

}

**Output**

meowing...

eating...

barking...

eating...

11. Write a java program to implement method overriding.

class Vehicle{

void run1(){

System.out.println("Vehicle is running");

}

}

class Bike2 extends Vehicle{

void run(){

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

}

public static void main(String args[]){

Bike2 obj = new Bike2();

obj.run();

//Vehicle v1 = new Vehicle();

obj.run1();

}

}

**Output**

Bike is running safely

Vehicle is running

12. Use eclipse or Netbean platform and acquaint with the various menus, create a test project, add a test class and run it see how you can use auto suggestions, auto fill. Try code formatter and code refactoring like renaming variables, methods and classes. Try debug step by step with a small program of about 10 to 15 lines which contains at least one if else condition and a for loop.

import java.util.Scanner;

// Creating Class

class Sample\_Program {

// main method

public static void main(String args[]) {

int i,count=0,n;

// creating scanner object

Scanner sc=new Scanner(System.in);

// get input number from user

System.out.print("Enter Any Number : ");

n=sc.nextInt();

// logic to check prime or not

for(i=1;i<=n;i++) {

if(n%i==0) {

count++;

}

}

if(count==2)

System.out.println(n+" is prime");

else

System.out.println(n+" is not prime");

}

}

Output:

Enter Any Number : 23

23 is prime

Enter Any Number : 45

45 is not prime

13. Write a Java program to demonstrate the OOP principles. [i.e., Encapsulation, Inheritance, Polymorphism and Abstraction]

/\* Encapsulation:

The fields of the class are private and accessed through getter and setter methods.\*/

class Person {

// private fields

private String name;

private int age;

// constructor

public Person(String name, int age) {

this.name = name;

this.age = age;

}

// getter and setter methods

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

/\* Abstraction:

The displayInfo() method provides a simple interface to interact with the object.\*/

public void displayInfo() {

System.out.println("Name: " + name);

System.out.println("Age: " + age);

}

}

/\* Inheritance:

Employee is a subclass of Person, inheriting its properties and methods.\*/

class Employee extends Person {

// private field

private double salary;

// constructor

public Employee(String name, int age, double salary) {

super(name, age);

this.salary = salary;

}

// getter and setter methods

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

/\* Polymorphism:

Overriding the displayInfo() method to provide a specific implementation for Employee.\*/

@Override

public void displayInfo() {

super.displayInfo();

System.out.println("Salary: " + salary);

}

}

public class OopPrinciplesDemo {

public static void main(String[] args) {

// Demonstrating encapsulation and abstraction

Person person = new Person("Madhu", 30);

System.out.println("Person Info:");

person.displayInfo();

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

// Demonstrating inheritance and polymorphism

Employee employee = new Employee("Naveen", 26, 50000);

System.out.println("Employee Info:");

employee.displayInfo();

}

}

Output:

Person Info :

Name: Madhu

Age: 30

====================

Employee Info :

Name : Naveen

Age : 26

Salary : 50000.0

14. Write a Java program to handle checked and unchecked exceptions. Also, demonstrate the usage of custom exceptions in real time scenario.

import java.io.File;

import java.io.FileReader;

import java.io.FileNotFoundException;

// Custom Exception

class InvalidAgeException extends Exception {

public InvalidAgeException(String message) {

super(message);

}

}

public class ExceptionsDemo {

// Method to demonstrate custom exception

public static void register(String name, int age) throws InvalidAgeException {

if (age < 18) {

throw new InvalidAgeException("User must be at least 18 years old.");

} else {

System.out.println("Registration successful for user: " + name);

}

}

public static void main(String[] args) {

//Handling Checked Exception

try {

File file = new File("myfile.txt");

// This line can throw FileNotFoundException

FileReader fr = new FileReader(file);

} catch (FileNotFoundException e) {

System.out.println("File not found: " + e.getMessage());

}

//Handling Unchecked Exception

try {

int[] arr = {1, 2, 3};

// Accessing an out-of-bound index

System.out.println(arr[6]);

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("Array index out of bounds: " + e.getMessage());

}

// Finally block to perform cleanup operations

finally {

System.out.println("Cleanup operations can be performed here.");

}

// Demonstrate custom exception handling

System.out.println("Demonstrating Custom Exception:");

try {

// Invalid age for registration

register("Madhu", 17);

} catch (InvalidAgeException e) {

System.out.println("Custom Exception Caught: " + e.getMessage());

}

}

}

Output:

File not found: myfile.txt (The system cannot find the file specified)

Array index out of bounds: 6

Cleanup operations can be performed here.

Demonstrating Custom Exception:

Custom Exception Caught: User must be at least 18 years old.

15. Write a Java program to demonstrate the working of different collection classes. [Use package structure to store multiple classes].

**ListExample.java**

package collections;

import java.util.ArrayList;

public class ListExample {

public static void main(String[] args) {

ArrayList<String> list = new ArrayList<>();

list.add("Apple");

list.add("Banana");

list.add("Orange");

// to display

System.out.println("List Example:");

for (String fruit : list) {

System.out.println(fruit);

}

}

}

**SetExample.java**

package collections;

import java.util.HashSet;

public class SetExample {

public static void main(String[] args) {

HashSet<String> set = new HashSet<>();

set.add("Apple");

set.add("Banana");

set.add("Orange");

set.add("Apple"); // This won't be added since sets don't allow duplicates

// To display

System.out.println("Set Example:");

for (String fruit : set) {

System.out.println(fruit);

}

}

}

**CollectionsDemo.java**

package collections;

public class CollectionsDemo {

public static void main(String[] args) {

ListExample.main(args);

SetExample.main(args);

}

}

**Output:**

javac -d . ListExample.java

javac -d . SetExample.java

javac -d . CollectionsDemo.java

java collections.CollectionsDemo

List Example:

Apple

Banana

Orange

Set Example:

Apple

Orange

Banana