PROGRAM NO 1.

Accept N numbers and find their sum. Check whether the sum is prime or

not.

public class SumPrimeCheck {

public static boolean isPrime(int num) {

if (num <= 1) {

return false;

}

for (int i = 2; i \* i <= num; i++) {

if (num % i == 0) {

return false;

}

}

return true;

}

}

public static void main(String[] args) {

Scanner sc= new Scanner(System.in);

// Accepting the value of N

System.out.print("Enter N: ");

int N = sc.nextInt();

// Accepting N numbers

System.out.println("Enter " + N + " numbers:");

int sum = 0;

for (int i = 0; i < N; i++) {

int num = sc.nextInt();

sum += num;

}

// Checking if the sum is prime or not

boolean isPrime = isPrime(sum);

// Displaying the result

System.out.println("Sum of the numbers: " + sum);

if (isPrime) {

System.out.println("The sum is a prime number.");

} else {

System.out.println("The sum is not a prime number.");

}

sc.close();

}

}

PROGRAM NO 2.

Evaluate the following series using switch statement

a) a + 2a/b + 3a/2b + . . . . . . . +na/(n-1)b

b) 1 + ½ + ¼ + 1/8 + . . . . . . ..

import java.util.Scanner;

public class SeriesEvaluation {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// Printing options for the user

System.out.println("Select the series to evaluate:");

System.out.println("1) Series 1");

System.out.println("2) Series 2");

System.out.print("Enter your choice: ");

int choice = sc.nextInt();

// Switch statement to handle user's choice

switch (choice) {

case 1:

// Input values for Series 1

System.out.print("Enter the value of a and b: ");

double a = sc.nextDouble();

double b = sc.nextDouble();

System.out.print("Enter the value of n: ");

int n = sc.nextInt();

// Evaluation of Series 1

double sum1= 0;

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

sum1+= i \* a / ((i - 1) \* b);

}

System.out.println("Sum of Series 1: " + sum1);

break;

case 2:

// Input number of terms for Series 2

System.out.print("Enter number of terms for Series 2: ");

int terms = sc.nextInt();

// Evaluation of Series 2

double sum2= 0;

for (int i = 0; i < terms; i++) {

sum2+= 1.0 / Math.pow(2, i);

}

System.out.println("Sum of Series 2: " + sum2);

break;

default:

// Handling invalid choice

System.out.println("Invalid choice!");

}

sc.close();

}

}

PROGRAM NO 3.

To read a string and the two index values (i and j). Extract the string from

ith position to jth position.

import java.util.Scanner;

public class SubstringExtraction {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// Read the input string

System.out.print("Enter a string: ");

String inputString = sc.nextLine();

// Read the indices i and j

System.out.print("Enter i: ");

int i = sc.nextInt();

System.out.print("Enter j: ");

int j = sc.nextInt();

// Validate the indices

if (i < 0 || j < 0 || i >= inputString.length() || j >= inputString.length() || i > j) {

System.out.println("Invalid indices!");

} else {

// Extract substring from ith position to jth position

String newSubstring= "";

for (int index = i; index <= j; index++) {

newSubstring+= inputString.charAt(index);

}

//Print the final substring

System.out.println("Substring from position " + i + " to position " + j + ": " + newSubstring);

}

sc.close();

}

}

PROGRAM NO 4.

Create a Java class called Complex with the following details and variables within it as (i) Real (ii) Imaginary. Develop a Java program to perform addition and subtraction of two complex numbers by using the method add() and subtract() respectively by passing object as parameter and display result using method display(). Initialize the real and imaginary values of the complex number using a parameterized constructor.

public class Complex {

private double real;

private double imaginary;

// Parameterized constructor

public Complex(double real, double imaginary) {

this.real = real;

this.imaginary = imaginary;

}

// Method to add two complex numbers

public Complex add(Complex other) {

double realPart = this.real + other.real;

double imaginaryPart = this.imaginary + other.imaginary;

return new Complex(realPart, imaginaryPart);

}

// Method to subtract two complex numbers

public Complex subtract(Complex other) {

double realPart = this.real - other.real;

double imaginaryPart = this.imaginary - other.imaginary;

return new Complex(realPart, imaginaryPart);

}

// Method to display the complex number

public void display() {

if (imaginary >= 0) {

System.out.println(real + " + " + imaginary + "i");

} else {

System.out.println(real + " - " + Math.abs(imaginary) + "i");

}

}

public static void main(String[] args) {

// Creating complex numbers

Complex num1 = new Complex(3, 4);

Complex num2 = new Complex(1, 2);

// Performing addition and displaying result

System.out.println("Addition Result:");

Complex sum = num1.add(num2);

sum.display();

// Performing subtraction and displaying result

System.out.println("Subtraction Result:");

Complex difference = num1.subtract(num2);

difference.display();

}

}

PROGRAM NO 5.

A class called MyTime, which models a time instance with private instance variables: hour: between 0 to 23, minute: between 0 to 59, constructor shall invoke the setTime() method to set the instance variable (setTime(int hour, int minute): It shall check if the given hour and minute are valid before setting the instance variables). define methods - getHour(), getMinute(), nextMinute()Update this instance to the next minute and return this instance. Take note that the nextMinute() of 23:59 is 00:00 nextHour() is similar to the above. Write the code for the MyTime class. Also write a test program (called TestMyTime) to test all the methods defined in the MyTime class.

class MyTime {

private int hour;

private int minute;

// Constructor

public MyTime(int hour, int minute) {

setTime(hour, minute);

}

// Method to set the time

public void setTime(int hour, int minute) {

if (hour >= 0 && hour <= 23 && minute >= 0 && minute <= 59) {

this.hour = hour;

this.minute = minute;

} else {

System.out.println("Invalid time!");

}

}

// Method to get the hour

public int getHour() {

return hour;

}

// Method to get the minute

public int getMinute() {

return minute;

}

// Method to update time to next minute

public void nextMinute() {

minute++;

if (minute == 60) {

minute = 0;

hour++;

if (hour == 24) {

hour = 0;

}

}

// Method to update time to next hour

public void nextHour() {

hour++;

if (hour == 24) {

hour = 0;

}

}

}

public class TestMyTime {

public static void main(String[] args) {

// Creating a MyTime object with valid initial values

MyTime time = new MyTime(12, 30);

// Testing getHour() and getMinute()

System.out.println("Current Time: " + time.getHour() + ":" + time.getMinute());

// Testing nextMinute() and nextHour()

time.nextMinute();

System.out.println("Next Minute: " + time.getHour() + ":" + time.getMinute());

time.nextHour();

System.out.println("Next Hour: " + time.getHour() + ":" + time.getMinute());

}

PROGRAM NO 6.

Assume that a bank maintains two kinds of accounts for customers, one called as savings account and the other as current account. Create a class Account that stores customer name, account number, and type of account. From this derive the classes CurrAcct and SavAcct to make them more specific to their requirements. The savings account provides compound interest and withdrawal facilities. The current account does not provide interest. Current account holders should also maintain a minimum balance (Rs 5000), and if the balance falls below this level, a service charge (Rs 100) is imposed. Include the necessary methods in order to achieve the following tasks:

1. Accept deposit from customer and update the balance.
2. Display the balance.
3. Compute and deposit interest.
4. Permit withdrawal and update the balance.
5. Check for the minimum balance (only for Current account), impose penalty if necessary, and update the balance.

import java.util.Scanner;

// Define Account class

class Account {

String name;

long accNumber;

String accType;

double balance;

// Constructor

Account(String n, long no, double bal, String type) {

name = n;

accNumber = no;

accType = type;

balance = bal;

}

// Method to deposit money

public void deposit(double amount) {

balance += amount;

System.out.println("Amount " + amount + " deposited.");

}

// Method to display account balance

public void displayBalance() {

System.out.println("Account Balance: " + balance);

}

// Method to withdraw money

void withdraw(double amount) {

if (balance < amount) {

System.out.println("Insufficient funds for withdrawal.");

return;

}

balance -= amount;

System.out.println("Withdrawal successful. Current balance: " + balance);

}

}

// Define CurrentAccount class which extends Account

class CurrentAccount extends Account {

// Constructor

CurrentAccount(String name, long accNumber, double bal) {

super(name, accNumber, bal, "Current");

}

// Method to withdraw money with additional checks

void withdraw(double amount) {

super.withdraw(amount); // Call superclass withdraw method

checkMin(); // Check minimum balance after a withdrawal

}

// Method to check minimum balance

public void checkMin() {

if (balance < 5000) {

balance -= 100;

System.out.println("Minimum balance not maintained. Service charge of Rs 100 imposed.");

}

}

}

// Define SavingsAccount class which extends Account

class SavingsAccount extends Account {

// Constructor

SavingsAccount(String name, long accNumber, double bal) {

super(name, accNumber, bal, "Savings");

}

// Method to calculate compound interest

void calculateInterest(double rate, double time) {

double compoundInterest = balance \* Math.pow((1 + rate / 100), time) - balance;

balance += compoundInterest;

System.out.println("Compound Interest of Rs " + compoundInterest + " added. Current balance: " + balance);

}

}

// Main class BankSystem

public class BankSystem {

// Method to handle operations for CurrentAccount

static void handleCurrentAccount(Scanner sc, Scanner scan) {

int choice;

System.out.print("Enter name: ");

String name = scan.nextLine(); // Read the name input

System.out.print("Enter account number: ");

long number = sc.nextLong(); // Read the account number input

System.out.print("Enter balance: ");

double bal = sc.nextDouble(); // Read the balance input

CurrentAccount obj = new CurrentAccount(name, number, bal);

// Menu for Current Account operations

do {

System.out.println("\nCurrent Account Menu:");

System.out.println("1. Deposit");

System.out.println("2. Withdraw");

System.out.println("3. Check Minimum Balance");

System.out.println("4. Display Balance");

System.out.println("0. Back to Account Selection");

System.out.print("Enter your choice: ");

choice = sc.nextInt();

// Switch case for handling user choices

switch (choice) {

case 1:

System.out.print("Enter amount to deposit: ");

double depositAmount = sc.nextDouble();

obj.deposit(depositAmount);

break;

case 2:

System.out.print("Enter amount to withdraw: ");

double withdrawAmount = sc.nextDouble();

obj.withdraw(withdrawAmount);

break;

case 3:

obj.checkMin();

break;

case 4:

obj.displayBalance();

break;

case 0:

System.out.println("Returning to Account Selection...");

break;

default:

System.out.println("Invalid choice! Please enter a valid option.");

}

} while (choice != 0);

}

// Method to handle operations for SavingsAccount

static void handleSavingsAccount(Scanner sc, Scanner scan) {

int choice;

System.out.print("Enter name: ");

String name = sc.nextLine();

System.out.print("Enter account number: ");

long number = scan.nextLong();

System.out.print("Enter balance: ");

double bal = sc.nextDouble();

SavingsAccount obj = new SavingsAccount(name, number, bal);

// Menu for Savings Account operations

do {

System.out.println("\nSavings Account Menu:");

System.out.println("1. Deposit");

System.out.println("2. Withdraw");

System.out.println("3. Calculate Interest");

System.out.println("4. Display Balance");

System.out.println("0. Back to Account Selection");

System.out.print("Enter your choice: ");

choice = sc.nextInt();

// Switch case for handling user choices

switch (choice) {

case 1:

System.out.print("Enter amount to deposit: ");

double depositAmount = sc.nextDouble();

obj.deposit(depositAmount);

break;

case 2:

System.out.print("Enter amount to withdraw: ");

double withdrawAmount = sc.nextDouble();

obj.withdraw(withdrawAmount);

break;

case 3:

System.out.print("Enter interest rate: ");

double interestRate = sc.nextDouble();

System.out.print("Enter time (in years): ");

double time = sc.nextDouble();

obj.calculateInterest(interestRate, time);

break;

case 4:

obj.displayBalance();

break;

case 0:

System.out.println("Returning to Account Selection...");

break;

default:

System.out.println("Invalid choice! Please enter a valid option.");

}

} while (choice != 0);

}

// Main method

public static void main(String[] args) {

/\*\*

\* Explanation for using two Scanner objects:

\* When combining nextInt()/nextLong() with nextLine(), it can cause issues where the nextLine() is skipped,

\* and the following statements are executed unexpectedly.

\*

\* This is because nextInt(), etc. leaves behind a \n (newline character) in the buffer.

\*

\* Using two Scanner objects is a workaround for this issue.

\*

\* Alternatively, an empty nextLine() statement can be added before reading a line of text:

\*

\* Example:

\* int someNumber = sc.nextInt();

\* sc.nextLine(); // Consumes the newline character left by nextInt()/nextLong()

\* String name = sc.nextLine();

\*/

Scanner sc = new Scanner(System.in);

Scanner scan = new Scanner(System.in);

int accChoice;

// Menu for selecting account type

do {

System.out.println("\nSelect Account:");

System.out.println("1. Savings Account");

System.out.println("2. Current Account");

System.out.println("0. Exit");

System.out.print("Enter your choice: ");

accChoice = sc.nextInt();

// Switch case for handling user choices

switch (accChoice) {

case 1:

handleSavingsAccount(sc, scan);

break;

case 2:

handleCurrentAccount(sc, scan);

break;

case 0:

System.out.println("Exiting the program. Goodbye!");

break;

default:

System.out.println("Invalid choice! Please enter a valid option.");

}

} while (accChoice != 0);

}

}

PROBLEM NO 7.  
  
 Derive subclass called Cylinder from the superclass Circle with member variable (height) of type double, public methods (getHeight(), getVolume(), getArea()) and its constructors(Cylinder(height, radius), Cylinder(height, radius,color)). Create the two instances of cylinder and print similar cylinders if the area, volume and color of cylinders are same. Demonstrate the code reuse and polymorphism properties of Object oriented programming by inheriting the constructors and methods of the base class.  
  
package problemnumberone;  
  
*//In Circle.java file*public class Circle  
{ double radius;  
 char color;  
  
 *// Constructor to initialize radius* Circle(double radius)  
 { this.radius=radius;  
 color='g';  
 }  
  
 *// Constructor to initialize both radius and color  
 // Constructor overloading* Circle(double radius,char color)  
 { this.radius=radius;  
 this.color=color;  
 }  
  
 *// Method to return the radius* double getRadius()  
 { return radius;  
 }  
  
 *// Method to return the area* double getArea()  
 { return (Math.*PI*\*radius\*radius);

*//instead we can use return(Math. PI\*Math. pow(radius,2)* }  
}  
  
package problemnumberone;  
  
*//In Cylinder.java file*public class Cylinder extends Circle {  
 double height;  
  
 *// Constructor to initialize height* Cylinder(double height) {  
 super(0.0, 'g');  
 this.height = height;  
 }  
  
 *// Constructor to initialize height and radius* Cylinder(double height, double radius)  
 { super(radius);  
 this.height=height;  
 }  
  
 *//Constructor ot initialize height,radius,color* Cylinder(double height,double radius,char color)  
 { super(radius,color);  
 this.height=height;  
 }  
  
 *//Method to get height* double getHeight()  
 { return height;  
 }  
  
 *//Method to get volume of Cylinder* double getVolume()  
 { return super.getArea()\*height;  
 }  
  
 *//Method to get Area of Cylinder* double getArea()  
 { return 2\*Math.*PI*\*radius\*(radius+height);  
 }  
  
 *//Method to check the similarity* void checkSimilar(Cylinder c1)  
 { *//Shortcut & operator is used instead we can use common & operator* if(getArea()==c1.getArea()&&getVolume()==c1.getVolume()&&color==c1.color)  
 System.*out*.println("The Cylinders are similar");  
 else  
 System.*out*.println("The Cylinders are not similar");  
 }  
  
}  
  
package problemnumberone;  
 import java.util.Scanner;  
  
*//In MainShape.java file  
//User entry program instead we can pass constants*public class MainShape  
{ public static void main(String[] args)  
 { double radius1,radius2,height1,height2;  
 char color1,color2;  
 Scanner ip=new Scanner(System.*in*);  
  
 System.*out*.println("Enter the radius for 1st cylinder:");  
 radius1=ip.nextDouble();  
 System.*out*.println("Enter the height for 1st cylinder:");  
 height1=ip.nextDouble();  
 System.*out*.println("Enter the color for 1st cylinder:");  
 color1=ip.next().charAt(0);  
  
 Cylinder c1=new Cylinder(height1,radius1,color1);  
  
 System.*out*.println("Enter the raidius for 2nd cylinder:");  
 radius2=ip.nextDouble();  
 System.*out*.println("Enter the height for 2nd cylinder:");  
 height2=ip.nextDouble();  
 System.*out*.println("Enter the color for 2nd cylinder:");  
 color2=ip.next().charAt(0);  
  
 Cylinder c2=new Cylinder(height2,radius2,color2);  
  
 c2.checkSimilar(c1);  
 }  
}  
  
  
  
  
PROBLEM NO 8.

Create an interface with name encryption and members as message and encrypt(). Derive two classes from this interface namely Nextchar and Prevchar In Nextchar class implement the method encrypt() to replace each character by its next character. In Prevchar class implement the method encrypt() to replace each character by its previous character.  
 For example: 1. If the input is “college” then the output is “dpmmfh” (replace each character by next characher).  
 If the input is zebra then the output is “ydaqz” (replace each character by previous character).  
  
  
package problemnumbertwo;  
  
public interface encryption  
{ final String *MESSAGE*="abc efg xyz";  
 void encrypt();  
}  
  
package problemnumbertwo;  
  
*//In NextChar.java file*public class NextChar implements encryption  
{  
 *//Overiding the funciton for finding the next char* public void encrypt()  
 {  
 char c;  
 char ch[]=new char[*MESSAGE*.length()];  
  
 for(int i=0;i<*MESSAGE*.length();i++)  
 { c=*MESSAGE*.charAt(i);  
  
 if(c=='z')  
 c='a';  
 else if(c=='Z')  
 c='A';  
 else if(c>='a'&&c<'z')  
 c++;  
 else if(c>='A'&&c<'Z')  
 c++;  
  
 ch[i]=c;  
 }  
  
 String s=new String(ch);  
 System.*out*.println("The Old String is :"+*MESSAGE*);  
 System.*out*.println("The next String is :"+s);  
  
 }  
}  
  
package problemnumbertwo;  
*//IN PreviousChar.java file*public class PreviousChar implements encryption  
{  
 *//Overiding the function for finding the previous char* public void encrypt()  
 {  
 char c;  
 char ch[] = new char[*MESSAGE*.length()];  
  
 for (int i = 0; i < *MESSAGE*.length(); i++)  
 {  
 c = *MESSAGE*.charAt(i);  
  
 if (c == 'a')  
 c = 'z';  
 else if (c == 'A')  
 c = 'Z';  
 else if (c > 'a' && c <= 'z')  
 c--;  
 else if (c > 'A' && c <= 'Z')  
 c--;  
  
 ch[i] = c;  
 }  
  
 String s = new String(ch);  
 System.*out*.println("The Old String is :" + *MESSAGE*);  
 System.*out*.println("The previous String is :" + s);  
  
 }  
}  
  
package problemnumbertwo;  
  
public class Mainencrypt  
{ public static void main(String[] args)  
 {  
 NextChar n=new NextChar();  
 n.encrypt();  
  
 PreviousChar p=new PreviousChar();  
 p.encrypt();  
 }  
}

Problem NO 9.

Create a package CIE which has two classes- Student and Internals. The class Student has members like usn, name, sem. The class internals has an array that stores the internal marks scored in six courses of the current semester of the student. Create another package SEE which has the class External which is a derived class of Student. This class has an array that stores the SEE marks scored in six courses of the current semester of the student. Import the two packages in a file that declares the final marks of N students in all six courses.

package CIE;  
import java.util.Scanner;  
  
*//pockage-CIE,class-Student*public class Student  
{ public String usn;  
 public String name;  
 public int sem;  
 Scanner ip=new Scanner(System.*in*);  
  
 *// Constructor to initialize usn,name,sem.* public Student()  
 { System.*out*.println("Enter the name of the student:");  
 name = ip.next();  
 System.*out*.println("Enter the usn of the student:");  
 usn = ip.next();  
 System.*out*.println("Enter the sem of the student:");  
 sem = ip.nextInt();  
 }  
}

package CIE;  
import java.util.Scanner;  
  
*//package-CIE,class-Internal*public class Internal extends Student  
{ public int[] internalmarks=new int[6];  
 Scanner ip=new Scanner(System.*in*);  
  
 *//Constructor to initialize marks of cie* public Internal()  
 { System.*out*.println("\*\*\*\*CIE EXAM\*\*\*\*");  
 for(int i=0;i<6;i++)  
 { System.*out*.print("Enter the marks of subject "+(i+1)+" :");  
 internalmarks[i] = ip.nextInt();  
 }  
 System.*out*.println();  
 }  
}

package SEE;  
import CIE.\*;  
import java.util.Scanner;  
  
*//package-SEE,class-External*public class External extends Internal  
{ int[] externalmarks=new int[6];  
 Scanner ip=new Scanner(System.*in*);  
  
 *//Constructor to initialize the marks of see* public External()  
 { System.*out*.println("\*\*\*\*SEE EXAM\*\*\*\*");  
 for(int i=0;i<6;i++)  
 { System.*out*.print("Enter the marks of subject "+(i+1)+" :");  
 externalmarks[i] = ip.nextInt();  
 }  
 System.*out*.println();  
 }  
  
 *//To display the student details and final marks* public void display()  
 {  
 System.*out*.println("Name :"+name);  
 System.*out*.println("USN :"+usn);  
 System.*out*.print("CIE Marks:");  
 for(int i=0;i<6;i++)  
 System.*out*.print(internalmarks[i]+" ");  
 System.*out*.println();  
 System.*out*.print("SEE Marks:");  
 for(int i=0;i<6;i++)  
 System.*out*.print(externalmarks[i]+" ");  
 System.*out*.println();  
 int final\_marks[]=new int[6];  
 for(int i=0;i<6;i++)  
 final\_marks[i]=(externalmarks[i]/2)+internalmarks[i];  
 System.*out*.print("Final Marks:");  
 for(int i=0;i<6;i++)  
 System.*out*.print(final\_marks[i]+" ");  
 System.*out*.println();  
 System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
 }  
}

package mainpackage;  
import SEE.\*;  
import java.util.Scanner;  
  
public class importingClass  
{ public static void main(String[] args)  
 { int n;  
 Scanner ip=new Scanner(System.*in*);  
  
 System.*out*.println("Enter the total number of student:");  
 n=ip.nextInt();  
  
 External[] e=new External[n];  
 for(int i=0;i<n;i++)  
 e[i]=new External();  
  
 System.*out*.println("\*\*\*\*\*\*Student Details\*\*\*\*\*\*\*");  
 for(int i=0;i<n;i++)  
 { System.*out*.println();  
 System.*out*.println();  
 System.*out*.println("\*\*\*Student "+(i+1)+" \*\*\*");  
 e[i].display();  
 }  
 }  
}

Problem NO 10

Write a java program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number

package sample;  
  
import java.util.Random;

import java.util.Scanner;  
  
*// Thread 1  
//class that implements the runnable interface*class Square implements Runnable   
{  
 int x;  
 Thread t;  
  
 *//constructor to initialize and call the start method* Square(int n)   
 {  
 x = n;  
 t = new Thread(this);  
 t.start();  
 }  
 *//override method from the runnable interface* public void run()   
 {  
 int sq = x \* x;  
 System.*out*.println("Square of " + x + " = " + sq);  
 }  
}  
  
*//Thread 2  
//class that extends the thread class*class cube extends Thread  
{  
 int x;  
   
 *//constructor that initialize and call the start method* cube(int n)   
 {  
 x = n;  
 start();  
 }  
  
 *//override method from the super class thread* public void run()   
 {  
 System.*out*.println("Cube of " + x + " = " + x \* x \* x);  
 }  
}  
  
*// Main thread*public class MainMultithread   
{  
 public static void main(String[] args)   
 { Scanner ip=new Scanner(System.in);

System.out.println("Enter the number random number" +  
 " need to be created:");  
 int n=ip.nextInt();

Random rand = new Random();  
 for (int i = 1; i <= n; i++)   
 {  
 int randnum = rand.nextInt(100);  
 System.*out*.println("Generated random number is " + randnum);  
 if (randnum % 2 == 0)  
 {  
 Square s = new Square(randnum);  
 }  
 else  
 {  
 cube c = new cube(randnum);  
 }  
   
 try  
 {  
 Thread.*sleep*(1000);  
 }   
 catch (InterruptedException ie)   
 {  
 System.*out*.println(ie);  
 }  
 }  
 }  
}

Problem NO 11

Write a java program to handle the following exceptions based on choice made by the user by writing suitable try and catch block.

i) ArithmeticException

ii) ArrayIndexOutOfBoundsException

iii) NumberFormatException

iv) StringIndexOutOfBoundException

v) NullPointerException

import java.util.Scanner;  
public class ExceptionClass  
{ public static void main(String[] args)  
 { int choice;  
 Scanner ip=new Scanner(System.*in*);  
 do  
 { System.*out*.println("\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*");  
 System.*out*.println("1--ArithmeticException");  
 System.*out*.println("2--ArrayIndexOutOfBoundsException");  
 System.*out*.println("3--NumberFormatException");  
 System.*out*.println("4--StringIndexOutOfBoundException");  
 System.*out*.println("5--NullPointerException");  
 System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
 System.*out*.println("Enter the choice:");  
 choice=ip.nextInt();  
  
 switch(choice)  
 { *//Arithmetic exception handling* case 1:  
 int num,den;  
  
 System.*out*.println("Enter the num and den");  
 num=ip.nextInt();  
 den=ip.nextInt();*//enter zero to get an exception* try  
 { System.*out*.println("The division of "+num+" by "  
 +den+" is:"+num/den);  
 }  
 catch(ArithmeticException e)  
 { System.*out*.println("Divide by zero exception");  
 }  
 break;  
  
 *// Array index out of bound exception* case 2:  
 int[] arr={1,2,3,4,5};  
  
 System.*out*.println("Enter the index to find the item" +  
 " in the array:");  
 int index=ip.nextInt();  
 *//Enter more than 4 to get an exception* try  
 { System.*out*.println("The item at given index" +  
 " is:"+arr[index]);  
 }  
 catch(ArrayIndexOutOfBoundsException e)  
 { System.*out*.println("Array index out of bound " +  
 "exception");  
 }  
 break;  
  
 *//Number format exception* case 3:  
 System.*out*.println("Enter the string to convert to"+  
 " integer:");  
 String s=ip.next();  
 *//enter letter or symbols to get an exception* try  
 { int x=Integer.*parseInt*(s);  
 System.*out*.println("The converted string " +  
 "to number is:"+x);  
 }  
 catch(NumberFormatException e)  
 { System.*out*.println("Number format exception");  
 }  
 break;  
  
 *// String index out of bound exception* case 4:  
 System.*out*.println("Enter the string:");  
 String s1=ip.next();  
 System.*out*.println("Enter the index to find the" +  
 " char at that index:");  
 int index2=ip.nextInt();  
 *//enter the index more than the string length* try  
 { char c=s1.charAt(index2);  
 System.*out*.println("The char is:"+c);  
 }  
 catch(StringIndexOutOfBoundsException e)  
 { System.*out*.println("String index out of" +  
 " bound exception");  
 }  
 break;  
  
 *//Null pointer exception* case 5:  
 String s2=null;

System.out.println("Enter the index to check the" +  
 " item:");  
 int itemind=ip.nextInt();

try  
 { char c1=s2.charAt(itemind);  
 System.*out*.println("The char at given index" +  
 " is:"+c1);  
 }  
 catch(NullPointerException e)  
 { System.*out*.println("Null pointer exception");  
 }  
 break;  
  
 case 6:  
 System.*out*.println("TATA BYE BYE");  
 break;  
  
 default:  
 System.*out*.println("Invalid choice entered");  
 }  
 }while(choice!=6);  
 }  
}

Problem No 12.

Define a class Sort with generic method by name Arrange(T[]) and Display(T[]).Write a program to sort array elements of different data types.

public class Sort  
{  
 *// Method to arrange (sort) the array* public static <T extends Comparable<T>> void Arrange(T[] arr)  
 {  
 int n = arr.length;  
 for (int i = 0; i < n - 1; i++)  
 { for (int j = 0; j < n - i - 1; j++)  
 { if (arr[j].compareTo(arr[j + 1]) > 0)  
 {  
 *// Swap arr[j] and arr[j+1]* T temp = arr[j];  
 arr[j] = arr[j + 1];  
 arr[j + 1] = temp;  
 }   
 }  
 }  
 }  
  
 *// Method to display the array* public static <T> void Display(T[] arr)  
 {  
 for (T element : arr)  
 {  
 System.*out*.print(element + " ");  
 }  
 System.*out*.println();  
 }  
  
 public static void main(String[] args)  
 {  
 Integer[] intArray = {5, 2, 8, 1, 3};  
 System.*out*.println("Original Integer Array:");  
 *Display*(intArray);  
 *Arrange*(intArray);  
 System.*out*.println("Sorted Integer Array:");  
 *Display*(intArray);

Float[] floatArray = {5.0f, 2.2f, 8.4f, 1.5f, 3.4f};  
 System.*out*.println("\nOriginal Float Array:");   
 *Display*(floatArray);  
 *Arrange*(floatArray);  
 System.*out*.println("Sorted Float Array:");  
 *Display*(floatArray);  
  
 Double[] doubleArray = {5.0, 2.2, 8.4, 1.5, 3.4};  
 System.*out*.println("\nOriginal double Array:");  
 *Display*(doubleArray);  
 *Arrange*(doubleArray);  
 System.*out*.println("Sorted double Array:");  
 *Display*(doubleArray);  
  
 Character[] charArray = {'z','p','h','l','o'};  
 System.*out*.println("\nOriginal char Array:");  
 *Display*(charArray);  
 *Arrange*(charArray);  
 System.*out*.println("Sorted chars Array:");  
 *Display*(charArray);

String[] stringArray = {"banana", "apple", "orange", "grape"};  
 System.*out*.println("\nOriginal String Array:");  
 *Display*(stringArray);  
 *Arrange*(stringArray);  
 System.*out*.println("Sorted String Array:");  
 *Display*(stringArray);  
 }  
}