1.

import java.util.\*;

import java.io.IOException;

import java.lang.\*;

public class Easy1 {

static void valid(String c) throws IOException

{

if(!c.equals("India"))

{

throw new IOException("IOException occur at differnt name");

}

else

System.out.println("same name");

}

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner s=new Scanner(System.in);

System.out.println("enter aadhar num:");

int aadharNum=s.nextInt();

s.nextLine();

System.out.println("enter name:");

String name=s.next();

System.out.println("enter city:");

String city=s.next();

System.out.println("enter state:");

String state=s.next();

System.out.println("enter country:");

String country=s.next();

try

{

valid(country);

}

catch(Exception e)

{

System.out.println(e);

}

}

}

Output:

enter aadhar num:

123

enter name:

div

enter city:

salem

enter state:

tn

enter country:

India

same name

2. import java.util.Scanner;

class DivException extends Exception {

public DivException(String msg) {

super(msg);

}

}

public class DivCalc {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

try {

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

int num = in.nextInt();

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

int den = in.nextInt();

if (den == 0) {

throw new DivException("Denominator cannot be zero!");

}

int res = num / den;

System.out.println("Result: " + res);

} catch (DivException e) {

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

}

}

}

3. import java.util.\*;

class NameException extends Exception {

public NameException(String msg) {

super(msg);

}

}public class Employee {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

try {

System.out.print("Enter Employee Name: ");

String name = in.nextLine();

if (!name.matches("[a-zA-Z]+")) {

throw new NameException("Name is Invalid! It should contain only characters.");

}

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

} catch (NameException e) {

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

}

}

}

4. import java.util.\*;

class InvalidUsernameException extends Exception {

public InvalidUsernameException(String msg) {

super(msg);

}

}

class InvalidPasswordException extends Exception {

public InvalidPasswordException(String msg) {

super(msg);

}

}

public class UserAuth {

public static void main(String[] args) {

if (args.length < 3) {

return;

}

String email = args[0];

String username = args[1];

String password = args[2];

try {

if (!username.matches("[a-zA-Z0-9\_]+")) {

throw new InvalidUsernameException("Invalid Username! Only letters, numbers, and underscores allowed.");

}

if (password.length() < 6) {

throw new InvalidPasswordException("Invalid Password! It must be at least 6 characters long.");

}

System.out.println("Email: " + email);

System.out.println("Username: " + username);

System.out.println("Password is valid.");

} catch (InvalidUsernameException | InvalidPasswordException e) {

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

}

}

}

5. import java.util.\*;

class CovidException extends Exception {

public CovidException(String msg) {

super(msg);

}

}

public class PatientDetails {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

try {

System.out.print("Enter Patient Name: ");

String name = in.nextLine();

System.out.print("Enter Patient Age: ");

int age = in.nextInt();

System.out.print("Enter Oxygen Level (%): ");

int oxyLevel = in.nextInt();

System.out.print("Enter HRCT Report Value: ");

int hrct = in.nextInt();

if (oxyLevel < 95 && hrct > 10) {

throw new CovidException("Patient is Covid Positive(+) and Needs to be Hospitalized");

}

System.out.println("Patient Details:");

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

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

System.out.println("Oxygen Level: " + oxyLevel + "%");

System.out.println("HRCT Report: " + hrct);

System.out.println("Patient is Stable.");

} catch (CovidException e) {

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

}

}

}

----------------------

Medium:

1. package exceptionnpackagee;

public class MedQ1exception {

public static void main(String[] args) {

try {

System.out.println(BookID(5, 3));

System.out.println(BookID(-1, 3));

} catch (IllegalArgumentException e) {

System.out.println(e.getMessage());

}

}

public static int BookID(int n, int p) {

if (n < 0 || p < 0) {

throw new IllegalArgumentException("n or p should not be negative.");

}

if (n == 0 && p == 0) {

throw new IllegalArgumentException("n and p should not be zero.");

}

return n + p;

}

}

2. package exceptionnpackagee;

import java.util.\*;

public class MedQ2exception {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

try {

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

if (!sc.hasNextDouble()) throw new InvalidInputException("Only numbers are allowed.");

double num1 = sc.nextDouble();

System.out.print("Enter operation (+, -, \*, /, ^): ");

char op = sc.next().charAt(0);

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

if (!sc.hasNextDouble()) throw new InvalidInputException("Only numbers are allowed.");

double num2 = sc.nextDouble();

double result = 0;

switch (op) {

case '+': result = Calculator.add(num1, num2); break;

case '-': result = Calculator.subtract(num1, num2); break;

case '\*': result = Calculator.multiply(num1, num2); break;

case '/': result = Calculator.divide(num1, num2); break;

case '^': result = Calculator.power(num1, num2); break;

default: throw new InvalidInputException("Invalid operation.");

}

System.out.println("Result: " + result);

} catch (InvalidInputException | DivisionByZeroException | InvalidMultiplicationException e) {

System.out.println("Error: " + e.getMessage());

} finally {

sc.close();

}

}

}

class InvalidInputException extends Exception {

public InvalidInputException(String message) {

super(message);

}

}

class DivisionByZeroException extends Exception {

public DivisionByZeroException(String message) {

super(message);

}

}

class InvalidMultiplicationException extends Exception {

public InvalidMultiplicationException(String message) {

super(message);

}

}

class Calculator {

public static double add(double a, double b) {

return a + b;

}

public static double subtract(double a, double b) {

return a - b;

}

public static double multiply(double a, double b) throws InvalidMultiplicationException {

if (a == 0 || b == 0 || a == 1 || b == 1) {

throw new InvalidMultiplicationException("Multiplier and multiplicand should not be 0 or 1.");

}

return a \* b;

}

public static double divide(double a, double b) throws DivisionByZeroException {

if (b == 0) {

throw new DivisionByZeroException("Denominator should not be zero.");

}

return a / b;

}

public static double power(double base, double ex) {

return Math.pow(base, ex);

}

}

3. package exceptionnpackagee;

import java.time.LocalDate;

import java.time.Period;

import java.util.Scanner;

class InvalidEmpNumberException extends Exception {

public InvalidEmpNumberException(String message) {

super(message);

}

}

class InvalidDateOfJoinException extends Exception {

public InvalidDateOfJoinException(String message) {

super(message);

}

}

class Employee {

private int empCode;

private String name;

private LocalDate dob;

private LocalDate doj;

public Employee(int empCode, String name, LocalDate dob, LocalDate doj) throws InvalidEmpNumberException, InvalidDateOfJoinException {

if (empCode <= 0) {

throw new InvalidEmpNumberException("Employee code must be a positive integer.");

}

if (!dob.isBefore(doj)) {

throw new InvalidDateOfJoinException("Date of Birth must be before Date of Appointment.");

}

this.empCode = empCode;

this.name = name;

this.dob = dob;

this.doj = doj;

}

public void displayDetails() {

int experience = Period.*between*(doj, LocalDate.*now*()).getYears();

System.***out***.println("Employee Details:");

System.***out***.println("Employee Code: " + empCode);

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

System.***out***.println("Date of Birth: " + dob);

System.***out***.println("Date of Appointment: " + doj);

System.***out***.println("Years of Experience: " + experience);

}

}

public class Med3 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.***in***);

try {

System.***out***.print("Enter Employee Code: ");

int empCode = scanner.nextInt();

scanner.nextLine();

System.***out***.print("Enter Employee Name: ");

String name = scanner.nextLine();

System.***out***.print("Enter Date of Birth (yyyy-mm-dd): ");

LocalDate dob = LocalDate.*parse*(scanner.nextLine());

System.***out***.print("Enter Date of Appointment (yyyy-mm-dd): ");

LocalDate doj = LocalDate.*parse*(scanner.nextLine());

Employee employee = new Employee(empCode, name, dob, doj);

employee.displayDetails();

} catch (InvalidEmpNumberException | InvalidDateOfJoinException e) {

System.***out***.println("Error: " + e.getMessage());

} catch (Exception e) {

System.***out***.println("Invalid input! Please enter details in the correct format.");

} finally {

scanner.close();

}

}

}

Output:

Enter Employee Code: 5632

Enter Employee Name: Div

Enter Date of Birth (yyyy-mm-dd): 2004-08-04

Enter Date of Appointment (yyyy-mm-dd): 2018-08-03

Employee Details:

Employee Code: 5632

Name: Div

Date of Birth: 2004-08-04

Date of Appointment: 2018-08-03

Years of Experience: 6

--------------------------

Hard:

1.

package exceptionnpackagee;

import java.time.LocalDate;

import java.time.Period;

import java.util.Scanner;

class InvalidEmpNumberException extends Exception {

public InvalidEmpNumberException(String message) {

super(message);

}

}

class InvalidDateOfJoinException extends Exception {

public InvalidDateOfJoinException(String message) {

super(message);

}

}

class Employee {

private int empCode;

private String name;

private LocalDate dob;

private LocalDate doj;

public Employee(int empCode, String name, LocalDate dob, LocalDate doj) throws InvalidEmpNumberException, InvalidDateOfJoinException {

if (empCode <= 0) {

throw new InvalidEmpNumberException("Employee code must be a positive integer.");

}

if (!dob.isBefore(doj)) {

throw new InvalidDateOfJoinException("Date of Birth must be before Date of Appointment.");

}

this.empCode = empCode;

this.name = name;

this.dob = dob;

this.doj = doj;

}

public void displayDetails() {

int experience = Period.*between*(doj, LocalDate.*now*()).getYears();

System.***out***.println("Employee Details:");

System.***out***.println("Employee Code: " + empCode);

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

System.***out***.println("Date of Birth: " + dob);

System.***out***.println("Date of Appointment: " + doj);

System.***out***.println("Years of Experience: " + experience);

}

}

public class Hard1 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.***in***);

try {

System.***out***.print("Enter Employee Code: ");

int empCode = scanner.nextInt();

scanner.nextLine();

System.***out***.print("Enter Employee Name: ");

String name = scanner.nextLine();

System.***out***.print("Enter Date of Birth (yyyy-mm-dd): ");

LocalDate dob = LocalDate.*parse*(scanner.nextLine());

System.***out***.print("Enter Date of Appointment (yyyy-mm-dd): ");

LocalDate doj = LocalDate.*parse*(scanner.nextLine());

Employee employee = new Employee(empCode, name, dob, doj);

employee.displayDetails();

} catch (InvalidEmpNumberException | InvalidDateOfJoinException e) {

System.***out***.println("Error: " + e.getMessage());

} catch (Exception e) {

System.***out***.println("Invalid input! Please enter details in the correct format.");

} finally {

scanner.close();

}

}

}

Output:

Enter Employee Code: 2563

Enter Employee Name: Karan

Enter Date of Birth (yyyy-mm-dd): 204-05-03

Invalid input! Please enter details in the correct format.

2.

package exceptionnpackagee;

import java.util.Arrays;

import java.util.Scanner;

class InvalidBattingAverageException extends Exception {

public InvalidBattingAverageException(String message) {

super(message);

}

}

class CricketPlayer {

private String name;

private int noOfInnings;

private int noTimesNotOut;

private int totalRuns;

private double batAvg;

public CricketPlayer(String name, int noOfInnings, int noTimesNotOut, int totalRuns) {

this.name = name;

this.noOfInnings = noOfInnings;

this.noTimesNotOut = noTimesNotOut;

this.totalRuns = totalRuns;

this.batAvg = 0;

}

public void calculateAverage() throws InvalidBattingAverageException {

if (noOfInnings - noTimesNotOut == 0) {

throw new InvalidBattingAverageException("Batting average cannot be calculated as the player was never out.");

}

this.batAvg = (double) totalRuns / (noOfInnings - noTimesNotOut);

}

public double getBatAvg() {

return batAvg;

}

public static void sortPlayers(CricketPlayer[] players) {

Arrays.*sort*(players, (p1, p2) -> Double.*compare*(p2.getBatAvg(), p1.getBatAvg()));

}

public void display() {

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

System.***out***.println("Innings Played: " + noOfInnings);

System.***out***.println("Times Not Out: " + noTimesNotOut);

System.***out***.println("Total Runs: " + totalRuns);

System.***out***.println("Batting Average: " + batAvg);

System.***out***.println("-------------------------");

}

}

public class Hard2 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.***in***);

System.***out***.print("Enter number of players: ");

int n = scanner.nextInt();

CricketPlayer[] players = new CricketPlayer[n];

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

System.***out***.println("Enter details for player " + (i + 1) + ":");

System.***out***.print("Name: ");

scanner.nextLine();

String name = scanner.nextLine();

System.***out***.print("Number of Innings: ");

int innings = scanner.nextInt();

System.***out***.print("Number of Times Not Out: ");

int notOut = scanner.nextInt();

System.***out***.print("Total Runs: ");

int runs = scanner.nextInt();

players[i] = new CricketPlayer(name, innings, notOut, runs);

try {

players[i].calculateAverage();

} catch (InvalidBattingAverageException e) {

System.***out***.println("Error: " + e.getMessage());

}

}

CricketPlayer.*sortPlayers*(players);

System.***out***.println("\nSorted Players Based on Batting Average:");

for (CricketPlayer player : players) {

player.display();

}

scanner.close();

}

}

Output:

Enter number of players: 1

Enter details for player 1:

Name: div

Number of Innings: 1

Number of Times Not Out: 2

Total Runs: 34

Sorted Players Based on Batting Average:

Player Name: div

Innings Played: 1

Times Not Out: 2

Total Runs: 34

Batting Average: -34.0