

**Fahrenheit to Celsius**

**Complete the static methods in the class Utility as per following requirements**

**Option 1 : Method fahrenheitToCelcius** :

This method should convert farhenheit in to celcius based on the formula [celcius =

(farhenheit - 32) X 5 / 9]

The method takes farhenheit(double) as input parameter

Method should return calculated temperature celcius rounded to an integer

**Option 2 : Method getLevel :**

Takes an integer array as input parameter

Should calculate the sum of all array elements and return a String as per below rules

HIGH - when sum is greater than or equal to 100, MEDIUM - when sum is greater than or

equal to 70, LOW - when sum is less than 70

**Complete the main method in class Source as below**

Program should take console input and call appropriate methods of Utility class based on the

Input.

Input and Output sample formats are given below in Example section

First input should be option 1 or 2.

Option 1 is for Celcius calculation

Option 2 for finding Level

In case of option 1, the second input should be temperature in farhenheit

In case of option 2, the second input should be number of elements in the array, followed by

the array elements In case of incorrect option, program should display 'Invalid Option'

Methods Of Utility Class:

public static String getLevel(int[] arr){}

public static int fahrenheitToCelcius(double farhenheit) {}

Example

Sample Input:

1 // Option

100 // temperature in Farhenheit

Expected Output:

38

Sample Input:

1

95.5

Expected Output:

35

Sample Input:

2 // option

3 // number of elements in array

40 // array elements

50 // array elements

11 // array elements

Expected Output:

HIGH

Sample Input:

2

4

10

20

5

10

Expected Output:

LOW

Solution:

import java.lang.Math;

import java.util.Scanner;

class Utility {

public static int fahrenheitToCelcius(double farhenheit) {

int celcius;

celcius=(int)Math.round(((farhenheit-32)\*5)/9);

return celcius;

}

public static String getLevel(int[] array) {

int sum=0,i;

for(i=0;i<array.length;i++)

{

sum+=array[i];

}

if(sum>=100)

return "HIGH";

else if(sum>=70 && sum<100)

return "MEDIUM";

else if(sum<70)

return "LOW";

else

return null;

//CODE END

}

}

public class Source {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

int choice=scan.nextInt();

switch(choice)

{

case 1:double farhenheit;

farhenheit=scan.nextDouble();

Utility u1=new Utility();

System.out.println(u1.fahrenheitToCelcius(farhenheit));

break;

case 2:int n,i;

n=scan.nextInt();

int ar[]=new int[n];

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

{

ar[i]=scan.nextInt();

}

Utility u2=new Utility();

System.out.println(u2.getLevel(ar));

break;

default:

System.out.println("Invalid Option");

break;

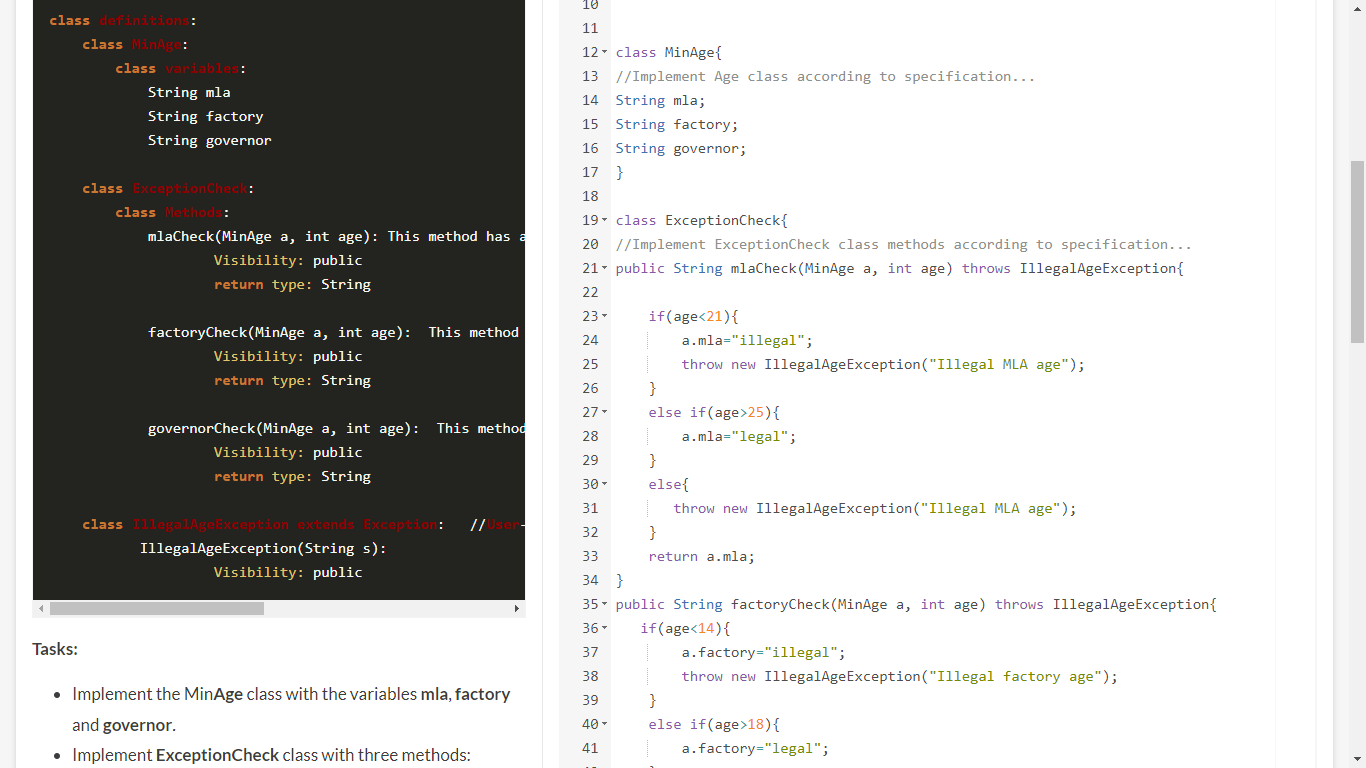
}

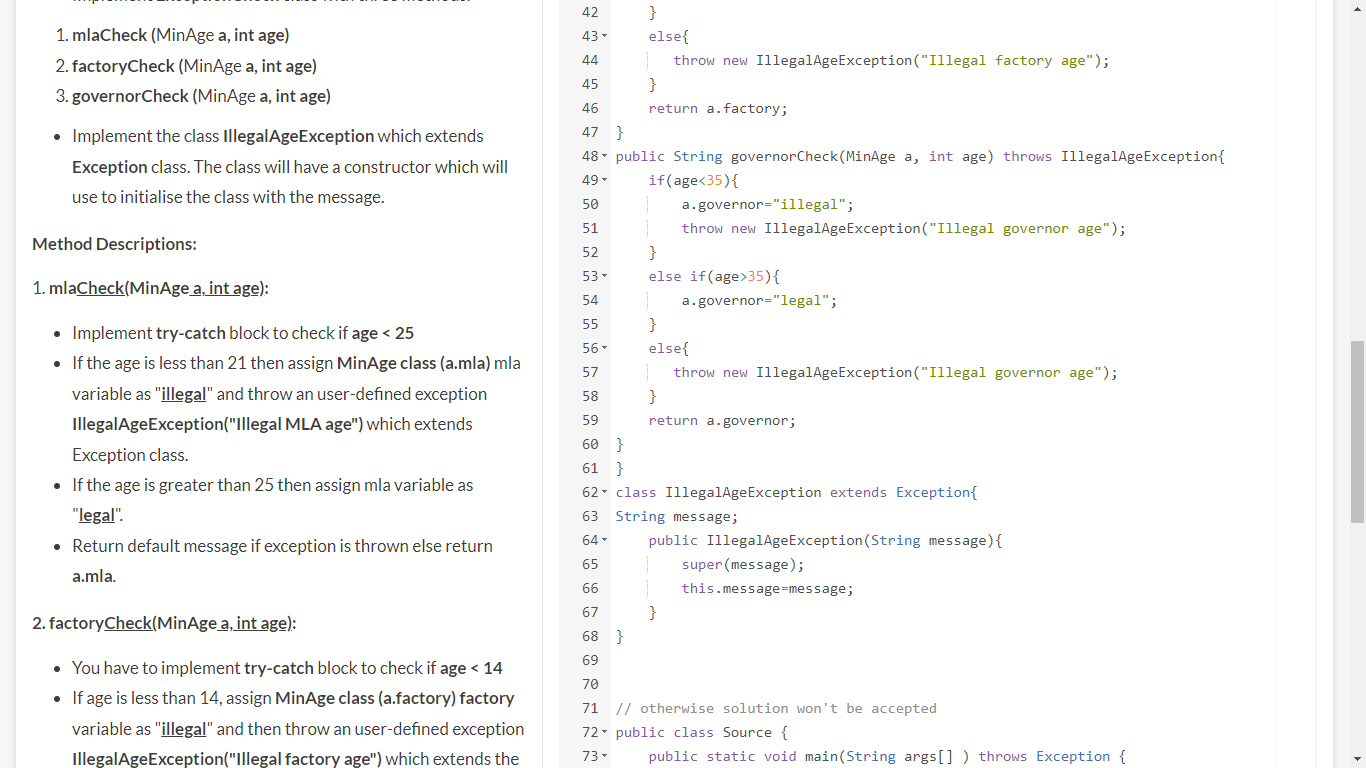
scan.close();

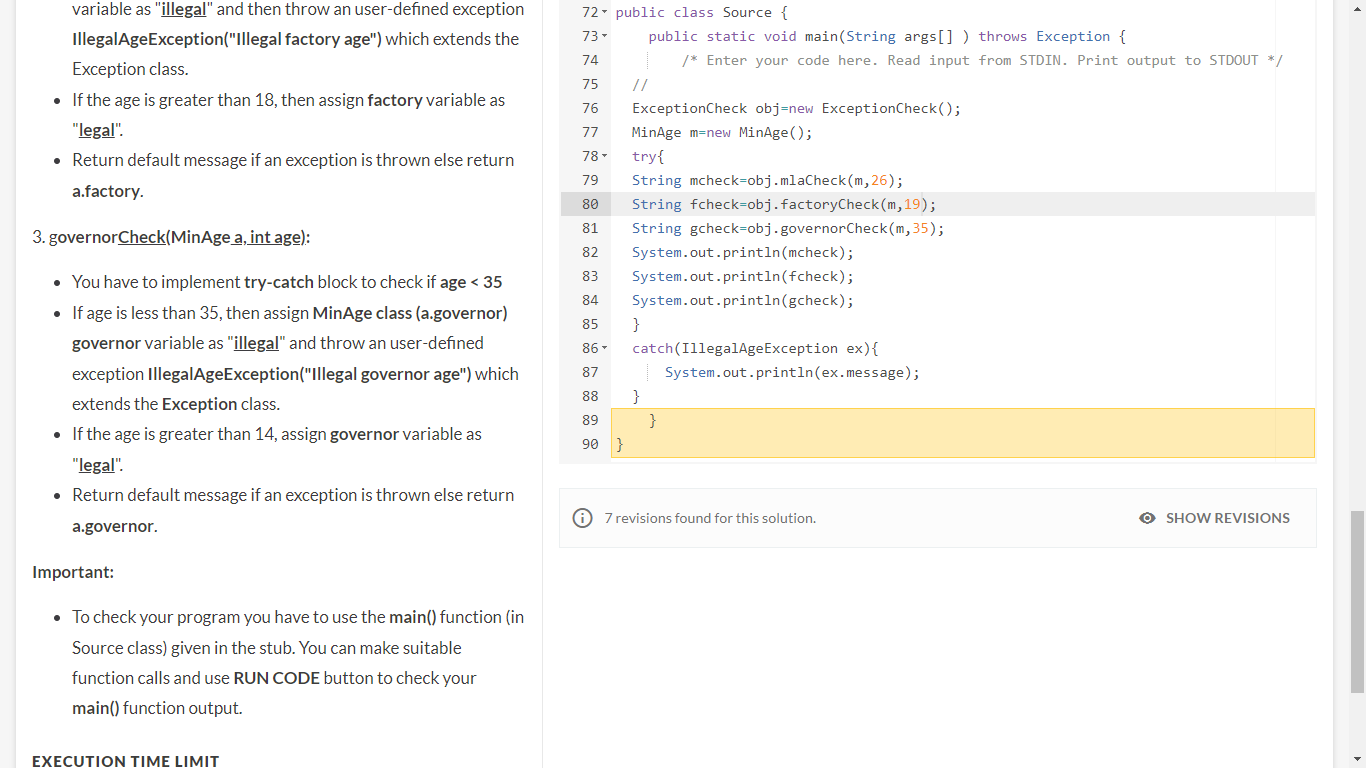
}

}

Exception Handling







Stream Api



Person p=personlst.stream()

.filter((e)->e.getPersonname().contains("e"))

.reduce((x,y)->x.getPersonage()<y.getPersonage()?x:y).get();

System.***out***.println(p.getPersonage());

Person x = personlst.stream()

.filter(e->e.getPersonname().contains("e"))

.min(Comparator.*comparingInt*(Person::getPersonage)).get();

**Count HR Designations**

Write a program which accepts designations in an String array.

Check the given array elements contain “HR” and display the total number of elements

which contains “HR” and display the elements in upper case also.

Display the message as given in Expected output.

**Example**:

Sample Input:

4

HRExecutive

Accountant

HRManager

SalesMan

Expected output:

Total 2 designations in HR Department

HREXECUTIVE

HRMANAGER

Sample Input:

2

Operator

Programmer

Expected output:

No designation of HR department found in given data

Sample Input:

-2

Operator

Programmer

Expected output:

INVALID INPUT

**Solution:**

import java.util.Scanner;

public class Source {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int no = scanner.nextInt();

if(no>0){

int count=0;

String designations[]=new String[no];

String input[] = new String[no];

for(int i = 0; i<no; i++) {

input[i] = scanner.next();

}

for(int i=0;i<no;i++){

if(input[i].contains("HR")){

designations[count]=input[i];

count++;

}

}

if(count>0)

{

System.out.println("Total "+ count +" designations in HR Department");

for(int j=0;j<count;j++)

{

System.out.println(designations[j].toUpperCase());

}

}

else

{

System.out.println("No designation of HR department found in given data");

}

}

else {

System.out.println("INVALID INPUT");

}

}

}

**Engineering Student**

Write code in given class **Student**which has private attributes:

**studentId : int**

**studentName : String**

**college: String**

with the method getDetails() to display the attributes.

Write code in another given class **EngineeringStudent**which should inherit **Student**class

and have to override **getDetails()** with additional information **stream**to be displayed.

**stream : String**

Accept the input from the user to create Student or EngineeringStudent object and display

the details using getDetails method. Other then "Student " or "EngineeringStudent" input, it

must display "Invalid Input".

### Example

Sample Input:

Student

111

Ram

National College

Expected output:

Student ID: 111

Student Name: Ram

College: National College

Sample Input:

EngineeringStudent

123

Ananya

IIT

Computer

Expected output:

Student ID: 123

Student Name: Ananya

College: IIT

Stream: Computer

Sample Input:

Trainer

Expected output:

Invalid Input

Solution:

import java.util.Scanner;

class Student {

private int studentId;

private String studentName;

private String college;

public Student(int studentId, String studentName, String college) {

super();

this.studentId = studentId;

this.studentName = studentName;

this.college = college;

}

public int getStudentId() {

return studentId;

}

public void setStudentId(int studentId) {

this.studentId = studentId;

}

public String getStudentName() {

return studentName;

}

public void setStudentName(String studentName) {

this.studentName = studentName;

}

public String getCollege() {

return college;

}

public void setCollege(String college) {

this.college = college;

}

public void getDetails() {

System.out.println("Student ID: " + studentId);

System.out.println("Student Name: " + studentName);

System.out.println("College: " + college);

}

}

class EngineeringStudent extends Student {

private String stream;

public EngineeringStudent(int studentId, String studentName, String college, String stream) {

super(studentId, studentName, college);

this.stream = stream;

}

public String getStream() {

return stream;

}

public void setStream(String stream) {

this.stream = stream;

}

public void getDetails() {

super.getDetails();

System.out.println("Stream: " + stream);

}

}

public class Source {

public static void main(String args[]) {

Scanner scanner = new Scanner(System.in);

String object = scanner.nextLine();

Student Student = null;

if (object.equals("EngineeringStudent")) {

int studentId = scanner.nextInt();

scanner.nextLine();

String studentName = scanner.next();

scanner.nextLine();

String college = scanner.nextLine();

String stream = scanner.nextLine();

Student=new EngineeringStudent(studentId, studentName, college, stream);

Student.getDetails();

} else if (object.equals("Student")){

int studentId = scanner.nextInt();

scanner.nextLine();

String studentName = scanner.next();

scanner.nextLine();

String college = scanner.nextLine();

Student = new Student(studentId, studentName, college);

Student.getDetails();

}else {

System.out.println("Invalid Input");

}

}

}

**Coach Player**

Create a java class Coach.Coach's job is to coach the Batsman and Bowler.

Implement loose coupling by introducing an Abstract class Player.

Create both the Batsman and Bowler classes, they must extend Player class.

Player class must have abstract method play() and returns void.

**play() : void**

Coach class has one private attribute:

**player : Player**

and two methods :-

**setPlayer(Player player) : void - will set Player**

**coach() : void - will call play() using player object**

will print “Batsman is batting”, if Batsman object is passed and “Bowler is bowling”, if Bowler Object is passed to the setPlayer method.Print "Invalid Input" if other then Batsman or Bowler is given as

**Sample Input .**

Pass the Batsman object or Bowler object to the setPlayer method and call the coach() method to print the information.

### Example

Sample Input:

Batsman

Expected output:

Batsman is batting

Sample Input:

Bowler

Expected output:

Bowler is bowling

Sample Input:

Umpire

Expected output:

Invalid Input

**Solution:**

import java.util.Scanner;

class Coach {

private IPlayer worker;

public void setWorker(IPlayer worker){

this.worker = worker;

}

public void coach() {

worker.play();

}

}

interface IPlayer

{

void play();

}

class Batsman implements IPlayer{

@Override

public void play() {

System.out.println("Batsman is batting");

}

}

class Bowler implements IPlayer{

@Override

public void play() {

System.out.println("Bowler is bowling");

}

}

public class Source {

public static void main(String[] args) {

// TODO Auto-generated method stub

Coach Coach = new Coach();

Scanner scanner = new Scanner(System.in);

String type = scanner.next();

if(type.equals("Batsman")) {

Coach.setWorker(new Batsman());

Coach.coach();

}else if(type.equals("Bowler")) {

Coach.setWorker(new Bowler());

Coach.coach();

} else {

System.out.println("Invalid Input");

}

}

}

StringTokenizer class

import java.util.StringTokenizer;

public class StringTokenizerExample {

public static int getSum(String s) {

int sum=0;

StringTokenizer st=new StringTokenizer(s);

while(st.hasMoreTokens()) {

int n=Integer.parseInt(st.nextToken());

sum+=n;

}

return sum;

}

public static void main(String[] args) {

int result=getSum("2 3 4 6");

System.out.println(result);

}

}

**Payments App**

Define a **abstract class named Payment** that contains a method named paymentDetails which should be declared abstract in this class,that outputs payment details w.r.t to CreditCardPayment and CashPayment.

Next, define a class named **CashPayment**that is derived from **Payment**. This class should redefine the paymentDetails method to indicate that the payment is in cash. Include appropriate constructor(s).

Define a class named**CreditCardPayment** that is derived from **Payment**. This class should contain instance variables for the name on the card, expiration date, and credit card number. Include appropriate constructor(s). Finally, redefine the **paymentDetails** method to include all credit card information in the printout.

In the main method create **CashPayment**and **CreditCardPayment** objects with different values and calls paymentDetails for each.

**Syntax for the classes required to build this Application**

1.CashPayment(double amount,String custname)---> Constructor for CashPayment

2.CreditCardPayment(double amount, String name, String expDate, String cardnumber)--->Constructor for CreditCardPayment

3.abstract class Payment

**Methods of Payment class**

**public abstract String paymentDetails()**---> this method returns the information depending on the type of Payment (Credit/CashPayment),it has to be redefined or oveririden in Credit/Cash classes and return the the reuqired sample output data mentioned below.

**Note:**

1.Payments made by credit card will have 5% discount on the amount

2.Payments made by cash will have 2% discount on the amount

3.Calculate and display the amount after applying discount.

**Sample Input 1:**

**1 ------------> Choice to be entered should be int type**

**10000 -------->total amount**

**Ajay -------->Customer Name**

**12/21 --------->Expiryd Date of the Credit Card**

**\*\*\*1234 --------->Credit Card Number**

**Sample Output 1:**

**Amount:9500.0CardNo:\*\*\*1234Validty:12/21Name:Ajay**

**Sample Input:2**

**2-------> Choice to be entered should be int type**

**6500-------> total amount**

**Ramesh------>customer name**

**Sample Output 2:**

**CashPayment:6370.0Customer:Ramesh**

**Note:**

**1.6370 is the amount after discount of 2%**

**2.9500.0** **is the amount after discount of 5%**

**. Code-Exception**

### 1. Create a Custom Exception "DivisionByZeroException" this exception should be thrown if a number is been divided by zero.

### 2.The code should also handle InputMismatchException ,if the input is not a numerical value that is read by the Scanner class.

### 3.The InputMismatchException is from java.util. package,need not to create this exception,its a predifined exception

### 4.If the inputs are right then it should display the result.

**Sample Input 1:**

10 -----------> Input 1

R -----------> Input 2

**Sample Output 1:**

**ERROR501**---->Note this message should be issued from the constructor of custom exception class

**Sample Input 2:**

10 -----------> Input 1

0 -----------> Input 2

**Sample Output 2:**

**ERROR502**

**Sample Input 3:**

10 -----------> Input 1

8 -----------> Input 2

**Sample Output 3:**

1.25