Your task here is to implement a**Java** code based on the following specifications. Note that your code should match the specifications in a precise manner. Consider **default visibility** of classes, data fields and methods are public unless mentioned otherwise.

**Specifications**

class definitions:

﻿class Product:

Data members:

﻿﻿ name: String

﻿ price: double

﻿ coupon: String

Define a parameterized constructor for all the data member.

class Validator:

﻿ validateCoupon(Product p)throws Exception

﻿return type: String

visibility: public

netPrice(Product p)

type: double

visibility: public

class InvalidCouponException:

﻿ method definitions:

﻿ InvalidCouponException(String msg)

visibility: public

**Task**

Class **Product**

**-**define the **String** variable **name**

**-**define the **double** variable **price**

**-**define the **String** variable **coupon**

-define a parameterized constructor for all the data members.

Class **Validator**

Implement the below methods for this class:

-**String** **validateCoupon(Product p):**

* **throw** an **InvalidCouponException** "Invalid Coupon" if the coupon is not valid. **The coupon** is valid if its name and discount value are separated with '-' and the discount value should be between 10-25(inclusive).

**Example:**

**name**= "**IPhone**" ; valid **coupons** are "**IPhone-10**", "**IPhone-20**", "**IPhone-18**" etc.

* **return** "Valid Coupon" if no exception found.

-**double netPrice(Product p):**

* **netPrice** = totalPrice-discountPrice.
* **return** netPrice if Coupon is valid else **return** totalPrice.

Class **InvalidCouponException**

* define custom exception class **InvalidCouponException** by **extending** the **Exception** class.
* define a parameterised constructor with a String argument to pass the message to the super class.

**Sample Input**

**Product** obj = new Product("IPhone",25000,"IPhone-10");

**Validator** val = new Validator();

**String** valCop = val.validCoupon(obj);

**double** price = val.netPrice(obj);

**Sample Output**

valCop = "Valid Coupon"

price = 22500.0

class Product {

//Write Your Code Here..

String name;

double price;

String coupon;

public Product (String name,double price,String coupon)

{

this.name=name;

this.price=price;

this.coupon=coupon;

}

}

class Validator{

//Write Your Code Here..

public String validateCoupon(Product p) throws InvalidCouponException{

String num=p.coupon.split("-")[1];

String name=p.coupon.split("-")[0];

int dis=Integer.parseInt(num);

if(name.equals(p.name)==false || (dis<10||dis>25))

throw new InvalidCouponException("Invalid Coupon");

return "Valid Coupon";

}

public double netPrice(Product p){

double discountPrice=0;

String name=p.coupon.split("-")[0];

int dis=Integer.parseInt(p.coupon.substring(p.coupon.length()-2));

discountPrice=p.price/100\*dis;

double netprice= p.price-discountPrice;

if(name.equals(p.name)==false||(dis<10||dis>25))

{

return p.price;

}

return netprice;

}

}

class InvalidCouponException extends Exception{

//Write Your Code Here..

public InvalidCouponException(String message)

{

super(message);

}

}

public class Source {

public static void main(String args[] ) throws Exception {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT \*/

}

}

Complete the classes using the Specifications given below. Consider default visibility of classes, data fields, and methods unless mentioned otherwise.

**Specifications**

class definitions:﻿

class Rating:

data members:

int imdbRating

int nominee

Rating(int imdbRating, int nominee): constructor **with** **public** visibility

**class** Validator:

method definitions:

canBeConsideredForTheAward(Rating rating) throws **Exception**:

**return** **type**: **String**

visibility: **public**

﻿

sendInvite(Rating rating) throws **Exception**:

**return** **type**: **String**

visibility: **public**

﻿

**class** MovieRatingException:

method definitions:

MovieRatingException(**String** msg)

visibility: **public** ﻿

**Task**

Class **Rating**

**-**define the **int**variable **imdbRating.**

**-**define the **int** variable **nominee**

**-**define a **constructor** according to the above specifications.

Class **Validator**

**Implement the below methods for this class:**

**-String** **canBeConsideredForTheAward(Rating rating) throws Exception:**

* Write a code to validate the criteria for getting the award.
* **throw a MovieRatingException**if **imdbRating** is less than **7**with the message "**Movie not eligible for Filmfare award**".
* **throw a MovieRatingException**if **nominee** is less than **4** with the message "**Minimum 4 nominee required**".
* If no above exception found then return a string message "**Considered for the award**".

**-String sendInvite(Rating rating):**

* Write a code to send an invite to the nominee.
* If **canBeConsideredForTheAward** method throws a **MovieRatingException** then return a message "**Not invited**".(Use try-catch block)
* If it throws any other exception then return a message "**other exception**".
* If no exception found then return a message "**Actors and Directors Invited**".

**Sample Input**

Rating rating = **new** Rating(9, 7);

Validator v = **new** Validator();

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

String s = v.canBeConsideredForTheAward(rating);

String t = v.sendInvite(rating);

s.toLowerCase();

t.toLowerCase();

**Sample Output**

considered **for** the award

actors **and** directors invited

class Rating {

//Your Code Goes Here..

int imdbRating;

int nominee;

public Rating(int imdbRating,int nominee){

this.imdbRating=imdbRating;

this.nominee=nominee;

}

}

class Validator {

//Your Code Goes Here..

public String canBeConsideredForTheAward(Rating rating) throws Exception{

if(rating.imdbRating<7){

throw new MovieRatingException("Movie not eligible for Filmfare award");

}

else if(rating.nominee<4) {

throw new MovieRatingException("Minimum 4 nominee required");

}

return "Considered for the award";

}

public String sendInvite(Rating rating) throws Exception{

try{

canBeConsideredForTheAward(rating);

}catch(MovieRatingException e){

return "Not invited";

}catch(Exception e){

return "other exception";

}

return "Actors and Directors Invited";

}

}

class MovieRatingException extends Exception{

public MovieRatingException(String msg){

super(msg);

}

}

public class Source {

public static void main(String args[] ) throws Exception {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT \*/

}

}

Your task here is to implement a Java code based on the following specifications. Note that your code should match the specifications in a precise manner. Consider **default visibility** of class unless mentioned otherwise.

Working in different sectors requires minimum qualification and age. In India we have age restriction for all types of employment which is governed by government of India.

**Specifications:**

**class** **definitions**:

**class** **MinAge**:

**class** **variables**:

            String mla

            String factory

            String governor

**class** **ExceptionCheck**:

**class** **Methods**:

            mlaCheck(MinAge a, int age): This method has an object  of MinAge **class** **and** **age**(**integer**) **as** **parameter**.

                    Visibility: public

**return** type: String

            factoryCheck(MinAge a, int age):  This method has an object of MinAge **class** **and** **age**(**integer**) **as** **the** **parameter**.

                    Visibility: public

**return** type: String

            governorCheck(MinAge a, int age):  This method has an object of MinAge **class** **and** **age**(**integer**) **as** **the** **parameter**.

                    Visibility: public

**return** type: String

**class** **IllegalAgeException** **extends** **Exception**:   //**User**-**Defined** **Exception** **class** **extends** **Exception** **class**

       IllegalAgeException(String s):

Visibility: public

**Tasks:**

* Implement the Min**Age** class with the variables **mla**, **factory** and **governor**.
* Implement **ExceptionCheck**class with three methods:

1. **mlaCheck (**MinAge**a, int age)**
2. **factoryCheck (**MinAge**a, int age)**
3. **governorCheck (**MinAge**a, int age)**

* Implement the class **IllegalAgeException**which extends **Exception** class. The class will have a constructor which will use to initialise the class with the message.

**Method Descriptions:**

1.**mlaCheck(MinAge a, int age):**

* Implement**try-catch** block to check if **age < 25**
* If the age is less than 21 then assign **MinAge class (a.mla)** mla variable as "**illegal**" and throw an user-defined exception **IllegalAgeException("Illegal MLA age")**which extends Exception class.
* If the age is greater than 25 then assign mla variable as "**legal**".
* Return default message if exception is thrown else return **a.mla**.

**2. factoryCheck(MinAge a, int age):**

* You have to implement**try-catch** block to check if **age < 14**
* If age is less than 14, assign **MinAge class (a.factory)** **factory** variable as "**illegal**" and then throw an user-defined exception **IllegalAgeException("Illegal factory age")**which extends the Exception class.
* If the age is greater than 18, then assign **factory** variable as "**legal**".
* Return default message if an exception is thrown else return **a.factory**.

3. g**overnorCheck(MinAge a, int age):**

* You have to implement **try-catch** block to check if **age < 35**
* If age is less than 35, then assign **MinAge class (a.governor)** **governor** variable as "**illegal**" and throw an user-defined exception **IllegalAgeException("Illegal governor age")**which extends the **Exception**class.
* If the age is greater than 14, assign **governor** variable as "**legal**".
* Return default message if an exception is thrown else return **a.governor**.

class MinAge{

String mla;

String factory;

String governor;

//Implement Age class according to specification...

}

class ExceptionCheck{

public String mlaCheck(MinAge a,int age){

if(age<25)

{

try{

if(age<21)

a.mla="illegal";

throw new IllegalAgeException("Illegal MLA age");

}

catch(IllegalAgeException e){

}return "Illegal MLA age";

}

else {

a.mla="legal";

}

return a.mla;

}

public String factoryCheck(MinAge a,int age){

if(age<14){

try{ a.factory="illegal";

throw new IllegalAgeException("Illegal factory age");

}

catch(IllegalAgeException e){

}

return "Illegal factory age";

}

else {

a.factory="legal";

}

return a.factory;

}

public String governorCheck(MinAge a,int age){

if(age<35){

try{ a.governor="illegal";

throw new IllegalAgeException("Illegal governor age");

}

catch(IllegalAgeException e){

}

return "Illegal governor age";

}

else {

a.governor="legal";

}

return a.governor;

}

}

//Implement ExceptionCheck class methods according to specification...

class IllegalAgeException extends Exception{

public IllegalAgeException(String s){

super(s);

}

}

// otherwise solution won't be accepted

public class Source {

public static void main(String args[] ) throws Exception {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT \*/

//

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*