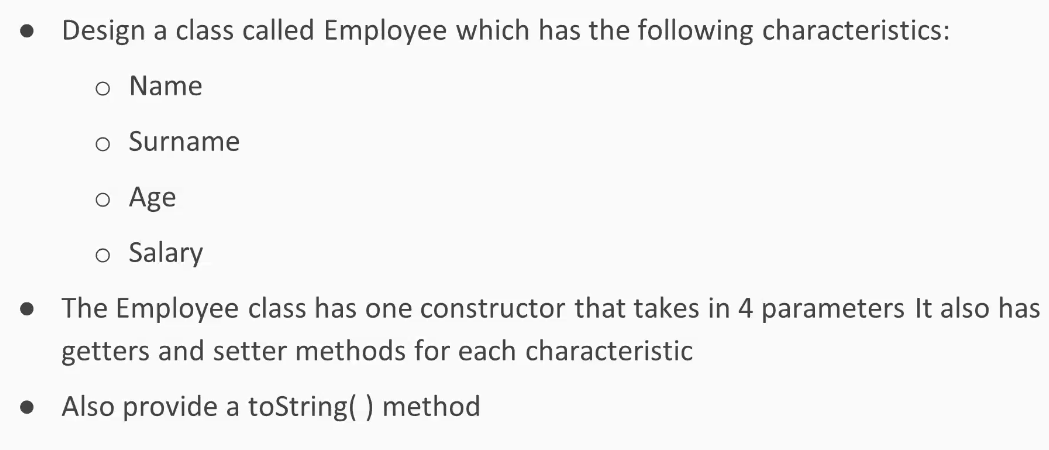
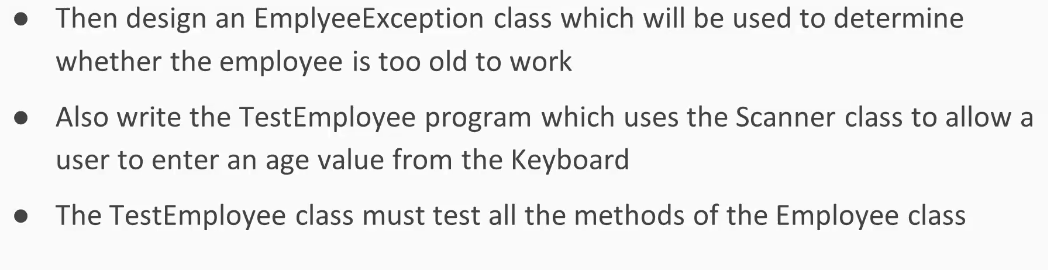
Hands On Exercises - Exception Handling

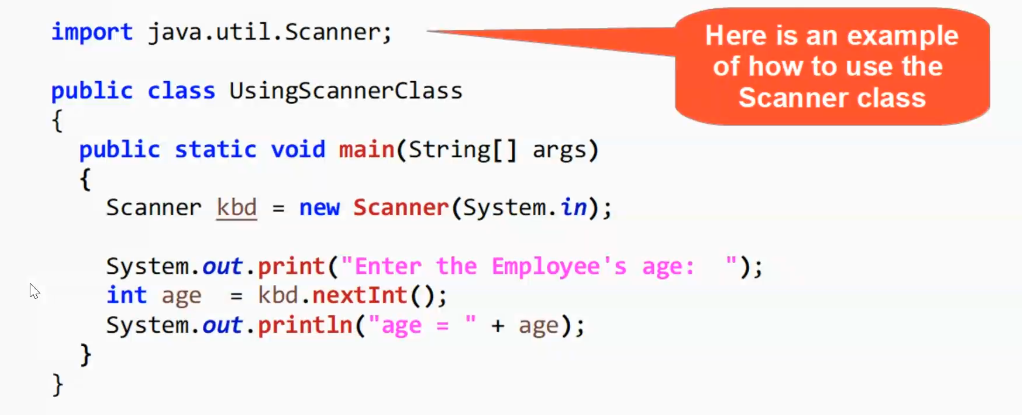
**Setup Instructions:**

1. Create a new Java Project called “JavaFundamentals4”.
2. Create your classes in the package “exceptions”
3. Define your classes as given below for each assignment.

Assignment 01



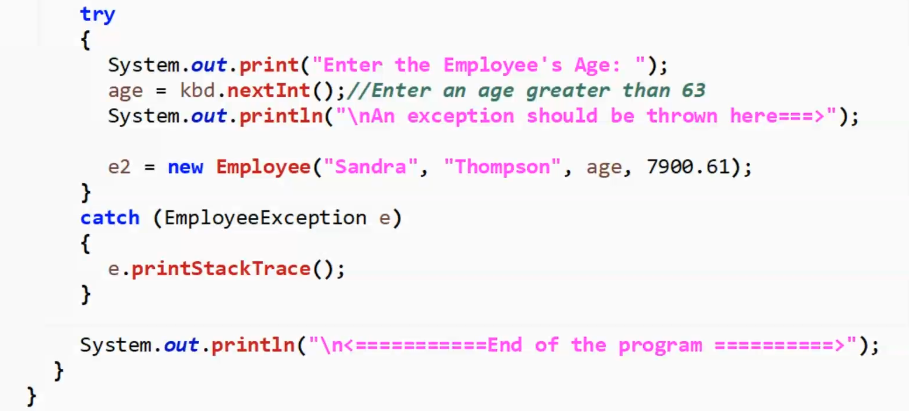
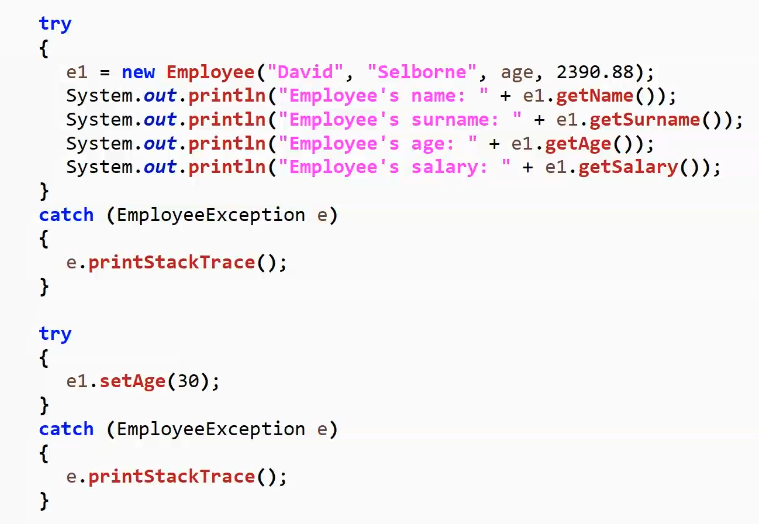
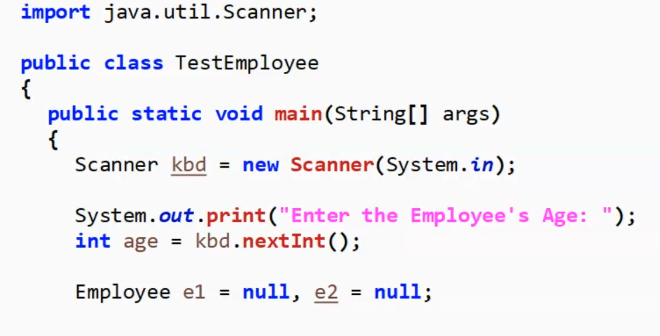




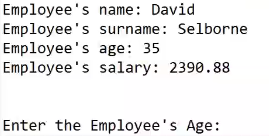
Instructions:

1. The Employee class constructor must check to see if the age>63.
2. If the age>63, throw a new EmployeeException passing in the String “Age cannot be older than 63.”
3. Set the Employee class instance variables equal to the arguments passed in to the constructor.
4. In the setter for the age variable, also check if the age>63 and if so, throw a new EmployeeException passing in the String “Age cannot be older than 63.”

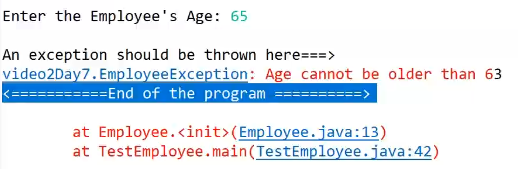
Use the below code to set up your TestEmployee class:



Run your TestEmployee program to test both the happy :) path



and the exception scenario:



Assignment 02

\*\*\* This is a doSelect Test Problem.

##### Handling Stuff

 Coding

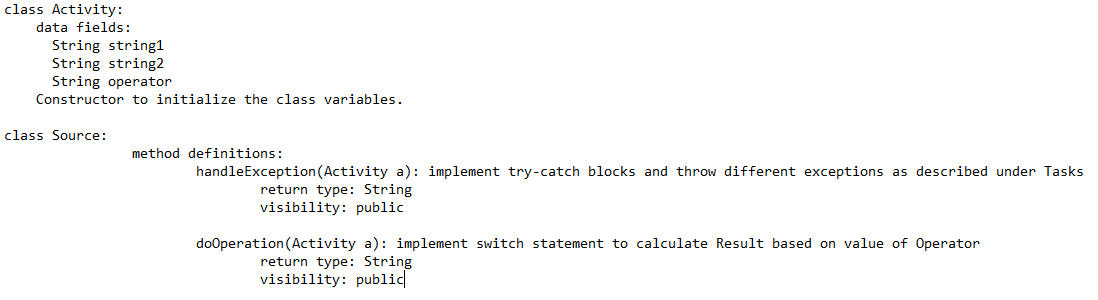
##### Description

In **Java**, we can use more than one catch block with the try block. Generally, multiple catch block is used to handle different types of exceptions, which means each catch block is used to handle different types of exceptions.

If you use multiple catch blocks for the same type of exception, then it will give you a compile-time error because **Java does not allow you to use multiple catch block for the same type of exception**. A catch block is always preceded by the try block.

Write a program to demonstrate Multiple Exceptions.

**Specifications:**



You have to implement the following methods under Source class:

* **handleException (Activity a)** - In this function you have to check for exceptions.
* **doOperation (Activity a)** - this function should implement the string operation between **string1** and **string2** for the operator **operator**.
* If **operator = '+'**, concat the strings **string1** and **string2**.
* **e.g.** for **string1 = "hello"**and **string2 = "world"**, then **result** = **"helloworld"**
* If **operator = '-'**, replace the contents of **string2** in **string1** with empty string.
* **e.g.** If **string1 = "helloworld"** and **string2 = "world"**, then **result = "hello"**

**Tasks:**

In the function **handleException** **(Activity a)**:

* Check that the value of either **string1** or **string2** variable is **null**, then throw appropriate exception for **NullPointerException** and return "**Null values found**".
* Check if the value of **operator** variable is not equal to these string operators ((+ or -) using logical AND operator. If the condition is true then throw and return the default exception with the Operator as the return message.
* If no exception is found return "**No Exception Found**".

In the function **doOperation (Activity a)**:

* perform the string operations, using switch statement and return the correct value.

**IMPORTANT:**

* If you want to test your program, you can implement a **main()** function given in the stub and you can use **RUN CODE** to test your main() provided you have made valid function calls with valid data required.

##### Execution time limit

10 seconds

REPORT AN ISSUE

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

class Activity{

//Implement Activity class here..

}

public class Source {

//Implement the two function given in description in here...

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

//Write your own main to check the program...

}

}

Assignment 03

\*\*\* This is a doSelect Test Problem.

##### Job Agency

 Coding

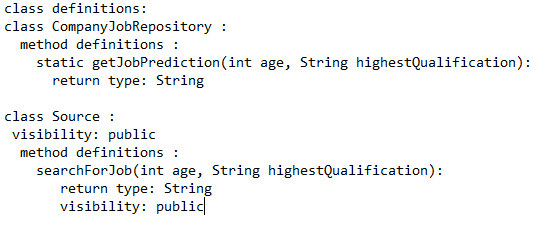
##### Description

**Case Study:**

You have to create business logic that simulates a job agency trying to search for openings at a company. The company has the following requirements from the candidates for their offerings:

* The candidate must be atleast 19 years of age. Otherwise, a NotEligibleException needs to be thrown with message "You are underage for any job"
* If the candidate is atleast 21 and the highest qualification is a B.E, then he/she is eligible for the role of a junior developer. In this case, a string needs to be returned as "We have openings for junior developer"
* The candidate is atleast 26 years of age and the highest qualification is an M.S or a PhD, then he/she is eligible for the role of a senior developer. In this case, a string needs to be returned as "We have openings for senior developer"
* If the candidate is atleast 19 years of age and the highest qualification is not any of B.E, M.S or PhD, then an exception named NotEligibleException needs to be thrown with the message "We do not have any job that matches your qualifications"
* For all other cases, a string needs to be returned as "Sorry we have no openings for now"
* You are supposed to create a class called CompanyJobRepository, which has a static method getJobPrediction() to meet the above requirements.

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 unless mentioned otherwise.



**Task**

On the basis of above case study implement the below classes and methods:

class **CompanyJobRepository**

-Implement the below methods for **CompanyJobRepository**

**-static String getJobPrediction(int age, String highestQualification):**

* Refer the case study above for the business logic

class **Source**

-Implement the below methods for Source

-**String** **searchForJob(int age, String highestQualification):**

* if age >= 200 or age <= 0, throw NotEligibleException with the message "The age entered is not typical for a human being"
* Otherwise, get the job predictions from CompanyJobRepository.
* You have to handle the NotEligibleException thrown by getJobPrediction(), in which case you have to return the message of the exception caught.

Class **NotEligibleException**

* Define custom exception class NotEligibleException by **extending** the **Exception** class.
* Define a parameterized constructor with a String argument to pass the message to the super class.

**NOTE**

* You can make suitable function calls and use **RUN CODE** button to check your **main()** method output.

##### Execution time limit

10 seconds

REPORT AN ISSUE

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

class CompanyJobRepository {

}

public class Source {

public static void main(String args[] ) {

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

}

}

class NotEligibleException extends Exception {

}

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