Employee Salary Simulation App

**Grade settings**: Maximum grade: 100  
**Run**: Yes **Evaluate**: Yes  
**Automatic grade**: Yes

**Important Instructions:**

●       Please read the document thoroughly before you code.

●       Import the given skeleton code into your Eclipse.

●       Do not change the Skeleton code or the package structure, method names, variable names, return types, exception clauses, access specifiers etc.

●       You can create any number of private methods inside the given class.

●       You are provided with a **Main class** with the main method to check the correctness of the test methods written.

●       Having completed writing the test methods, execute the main method and identify the result.

**Assessment Coverage:**

●        JUnit annotations & **exception handling** rules

You are required to write Junit test case and check the correctness of the application developed.

**Functional Requirements:**

The application has the below classes and methods implemented.

You are provided with a model class Employee

**Component Specification : Employee (Model Class)**

|  |  |  |
| --- | --- | --- |
| **Class** | **Attribute(s)** | **Template Method(s)** |
| Employee | String name  double  monthlySalary  int age | Necessary getters, setters and constructors provided as part of skeleton with Three argument constructor and Also the equals , hashCode and toString methods are overridden. |

Here the

Name -> for assigning employee name.

MonthlySalary -> what is the salary of employee per month

Age ->  what is the age of employee.

**Component Specification:  InvalidSalaryException**

**(This class inherits the Exception Class)**

|  |  |
| --- | --- |
| Type(Class) | Methods |
| InvalidSalaryException | Provided with a single argument constructor InvalidSalaryException  (String message) |

**Component Specification : EmpService (Utility Class)**

**Business Rule:**

The below is the requirement implemented in the Utility class for which JUnit test cases are to be written and tested.

|  |  |  |
| --- | --- | --- |
| **Class** | **Method(s)** | **Responsibilities** |
| EmpService | double calculateYearlySalary(Employee employee) | If the employee parameter matches it will calculate Yearly Salary and it will return double salary. if employee salary is invalid it will throw InvalidSalaryException with message “Provide valid salary”. |
| EmpService | double calculateAppraisal(Employee employee) | If the employee parameter matches it will check salary less than Rs 10000/- or not if  less than Rs 10000/- Yearly appraisal will provide Rs 500/-  and the salary is greater than or equal to  Rs 10000/- it will provide Rs 1000/- appraisal  if employee salary is invalid it will throw InvalidSalaryException with message “Provide valid salary”. |
| EmpService | double calculateYearlySalaryAfterAppraisal(Employee employee) | If the employee parameter matches it will calculate total salary after appraisal applied and it will give total annual salary after increment. if employee salary is invalid it will throw InvalidSalaryException with message “Provide valid salary”. |
| EmpService | public boolean validateSalary(double salary) | If the salary is not greater than zero then this method will throw an exception “InvalidSalaryException” with the message “Provide valid salary”. Otherwise it will return true |

**Testing Scenarios:**

You are provided with a class “**TestEmployee**” to do this testing.

**Note :**

The below are the **test methods** to be implemented in **TestEmployee** class.

|  |  |
| --- | --- |
| **Test Cases / Methods** | **Scenarios / Responsibilities** |
| test11ValidateSalaryForValid() | This method should verify the validateSalary(double salary) method  for the valid salary. Provide the valid  salary  and check the method |
| test12ValidateSalaryForInValid() | This method should verify the validateSalary(double salary) method  for the invalid salary. Provide the invalid  salary  and check the method |
| test13CalculateAppriasal() | This method should verify the **calculateAppraisal()** method call, whether it has invoked **only once** for valid employee with less than 10000/- salary and no exceptions thrown |
| test14CalculateAppriasal() | This method should verify the calculateAppraisal() method call, whether it has invoked only once for valid employee with greater than 10000/- salary and no exceptions thrown |
| test15CalulateApprisalForInvalidSalary() | This method should verify the calculateAppraisal() method call, whether it has been invoked only once for invalid employee salary and it will generate a message with “Provide  valid salary”. |
| test16CalculateYearlySalaryForValid() | This method should verify the calculateYearlySalary() method call, whether it has been invoked only once for valid employee salary and it will calculate the yearly salary before appraisal. |
| test17CalculateYearlySalaryForInValid() | This method should verify the calculateYearlySalary() method call, whether it has been invoked only once for invalid employee salary and exception will be thrown with the message “Provide valid salary” |
| test18CalculateYearlySalaryAfterAppraisal() | This method should verify the calculateYearlySalaryAfterAppraisal() method call, whether it has been invoked only once for valid employee salary  which is lesser than 10000 and it will calculate the yearly salary after appraisal. |
| test19CalculateYearlySalaryAfterAppraisal() | This method should verify the calculateYearlySalaryAfterAppraisal() method call, whether it has been invoked only once for valid employee salary  which is greater than or equal to 10000 and it will calculate the yearly salary after appraisal. |
| test20CalculateYearlySalaryAfterAppraisal() | This method should verify the calculateYearlySalaryAfterAppraisal() method call with the invalid salary ie <=0 . This will throw an exception with the message “Provide valid salary” |

Implement the test methods and provide the needed annotation to all the methods in TestEmployee class.

Also provide the required annotation, so that the test methods are executed in ascending order of the test method names.

You are provided with a Main class with the main method to check the correctness of the test methods written in TestEmployee class.

Having completed writing the test methods, uncomment the code in Main class and execute the main method.