Cebu Institute of Technology – University

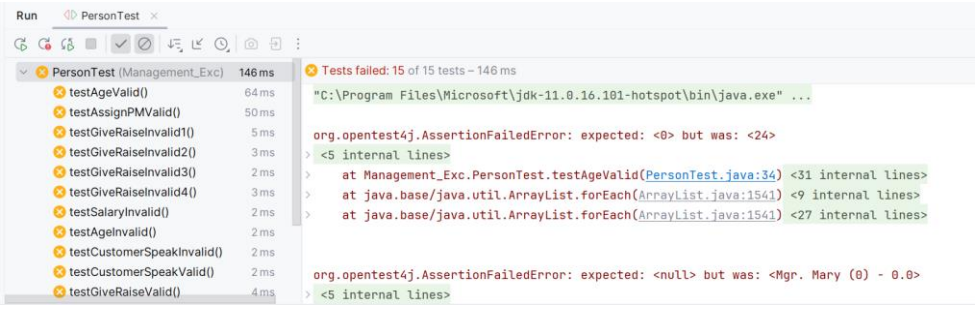
CSIT227 Object-Oriented Programming

Lab Activity 4.2 – Exceptions

Open IntelliJ IDEA, and clone the following repository: https://github.com/jdserato/CSIT227-LA4-2.git

As you have your own copy of the codebase, you will encounter an error in the PersonTest.java's import stateent: org.junit. You can either Show Context Actions in the org.junit and select "import junit5.8.1" or go to Project Structure, then Modules, then Dependencies, then Add Library > From Maven, and search for the following: **org.junit.jupiter** and choose the **5.8.1** version.

To run your program, use the PersonTest.java and run it. It should look like this:



There are 15 testcases, each testing different scenarios.

To submit the activity, just show the instructor the accomplished 15 testcases and I will record them manually.

Implement the following classes based on the following description.

A Person could either be an Employee or Customer. An Employee could be a Manager, a Developer, or just a regular Employee.

The Classes

Person

All Persons have the following private fields: a name and an age. Declare name as final since it would not ever be changed after constructing. Create a public getter for both. All Persons also celebrate birthday()s that prints "Happy birthday, name!" and they grow one year older. They could also performTask() that returns nothing but its implementation is to be performed on its subclasses. Whenever a Person is printed, it will print the name and the age enclosed in parenthesis, i.e., "name (age)".

For the age, make sure that it is valid. If it is negative, throw an IllegalArgumentException with message "Age must be non-negative"

Employee

The Employee is a Person that has the following field: double salary. Create a public getter and a protected setter for this field. The Employee's performTask() will simply print the following: name " is working". Whenever an Employee is printed, it would also have its salary listed after, i.e., "name (age) - salary", and there is no particular formatting for the salary's decimal places. As we attempt to promote reusability, try not to implement the same printing of the name and age again, but instead use the concept of inheritance, similar to the other tasks in this problem.

For the salary, make sure that it cannot go lower than 30,000. If it does, throw an IllegalArgumentException with message "Salary must be greater than or equal to 30000"

Manager

A Manager is an Employee that shall have a public method giveRaise(Employee, double) given the Employee and the increase that shall increase the Employee's current salary with the increase. Half of that increase shall also be added to the Manager's own salary. If the Employee in the argument is the Manager itself, then only the first one shall apply. For the giveRaise method, you must use the setter to set the Employee's salary. The said method must check the increase for validation such that it would not process negative values and shall instead print "Invalid increase". Additionally, when the Manager object is printed, it will have to be prepended by the abbreviation "Mgr."

For the raise in the giveRaise, make sure that it is valid. If it is negative, throw an IllegalArgumentException with message "Raise must be non-negative"

Developer

A Developer is an Employee that has a Manager projectManager. Create a public getter and protected setter for this field. The setter must only set the project manager when there is none, otherwise throw exception name " already has a manager". The Developer also has a method removePM() where it unassigns the developer's project manager. When a Developer has his birthday, the projectManager, if any, must also give him a raise amounting to 5% of the Developer's current salary. This must use the getter method of the salary; if there is no project manager, the Developer would not get a raise. Additionally, the performTask of the Developer must be changed to name "is coding". Additionally, when the Developer object is printed and it has a project manager, it will also output the name of the project manager inside square brackets i.e., "name (age) - salary [PM name]".

For the setProjectManager as mentioned above, when the developer already has a manager, throw an IllegalStateException with message “name already has a manager: managername”

Customer

A Customer is a Person that will print name " is browsing through" in its performTask method. It also has a method speak(Employee) where it will print "Oh, hello, [employee name]. Can you assist me?", except when the Employee is a Developer and the Customer's age is more than the Developer's age, in which case it will instead ask "Can I see your manager [name of manager]?"

Additionally, implement the static methods in Main. All methods will have the list of persons and a series of other parameters including strings for the names of the persons involved.

When the name is found and it is not the desired type, throw a ClassCastException with the message “name is not a/n type” where type is the desired type.

When the name is not at all found in the list of persons, throw a NoSuchElementException with the message “name does not exist”