4. **Consider the Payroll application code in Payroll.java. Modify the code so that it is free from** syntactical errors. Identify and list the users from the application and write their functionalities that is represented in the code. Draw Use-Cases. For the Use-Cases listed identify input and output variables. Prepare a test matrix that covers all the Equivalence Classes and Boundary Values for functional test validations. Now, further modify the code so that all the test cases of the test matrix have Assertions using JUnit testing tool. Design a Test file that accommodates these assertions and analyse the results.

Test Suit 1: Variable 1<<Details>>	As per Equivalence Class		As per Junit	Test Status (Pass/Fail)
Test Case ID	Input Value	**Expected Output**	Actual Output(as per JUNIT)	
1				
2				
N				
Test Suit 2: Variable 2<<Details>>				
Test Case ID		Expected Output	Actual Output	
1				
2				
N				

Enhance your learning:

- ✓ What is Parameterised Testing in JUnit? Write 6-7 lines of description regarding its process.
- ✓ Select any one use-case scenario from the above application under test and perform Parameterised Testing in JUnit.
- ✓ Suggest an alternative approach to Parameterised Testing that is supported by object oriented languages like C++ and Java.

Solution Must Contain: Revised code with Line Numbers, Code for Test Methods with Assertion Test Case Table with input and output classes, and test values. Screenshots test results and analysis description.