Wave Break Media Coding Exercise

Assumptions:

* As part of the requirements the error from rounding is in the range of 0.5 currency unit. This makes me assume that the rounding accuracy from using doubles will be sufficient and no need to use BigDecimal. (The loss of accuracy in money operations when using doubles comes from the machine representation of numbers as power of 2. This flaw can be overcame by using BigDecimal class.).
* The pension rate is in percentages, so no need to enter the % symbol.
* White space symbols are not present in the employee’s input data format and they will be treated as wrong format entry.
* The payslip’s period could be entered in the following formats, regardless the letters case :
* Name of the month;
* Number of the month;

Implementation:

* I have chosen to show you two approaches to implementing the coding exercise:Without software architecture pattern (SWP) - PayRollApp
* With Model-View-Presenter software architecture pattern – MvpPayRollApp.

1. **PayRollApp** consists from the following classes:

* PayRoll – entry point for the app. Handles the interaction with the user and the corresponding actions.
* Employee – POJO + static factory method for generating an Employee object. Validation of the input data format as well.
* PaySlip - pay slip model + static factory method for generating PaySlip object.
* TaxManager - the tax intelligence. Contains methods for calculating the required fields for the PaySlip.
* Utils – validation for the period format.
* EmployeeTest – JUnit test case. It contains six use cases covering one correct data input format and five incorrect. The objective here is to test the correct handling of incorrect user input without the program breaking.
* PaySlipBoundaryTest – JUnit test case. Tests the boundary use-cases for the different tax ranges.
* PayRollTest – Junit test case. Tests user input correct handling.
* JUnitTestSuite – an empty class, to define the test suite annotations.
* PayRollTestRunner – runs the above test case classes altogether automatically.

Testing PayRollApp - to test the application I would suggest any of these two options:

* The GitHub provided Java project can be imported in Eclipse as JavaProject and then can be ran there. For the PayRollApp the entry point is PayRoll run it by Run as -> Java Application. For the JUnit tests to run all tests automatically the entry point is PayRollTestRunner – Run as -> Java Application. The single test cases (EmployeeTest.java, PaySlipBoundaryTest.java, PayRollTest.java ) can be ran individually by Run as -> JUnit Test.
* The provided .jar files can be ran from the command prompt using JRE commands: java –jar payroll.jar for the application itself and java –jar payrolltest.jar for the test harness.

1. **MvpPayRollApp** Implementation.

MVP software pattern is derivative of MVC. Suitable for android applications, where decoupling of the view from the logic and the data model can be vital for not losing the current state when handling configuration changes. For this particular problem using this pattern is a bit of an overkill, but I wanted to demonstrate a starting point to elaborate to an Android app.

The Model layer is represented by PayRollView.java . All user interactions and writes to the console are done only from this class ( via interface method) in order to simulate only view purpose implementation. In the Android platform this will be any Activity class.

Presenter layer is represented by the Presenter.java class. This is the mediator between the view and the data/model. The presenter gets the user input from the view and passes it to the model layer in order to initiate population of the model’s data, initiate the generation of the PaySlip by the model and initiate viewing of the payslip in the view layer. All interactions between the layers happens via interfaces. In this package logically belongs the TaxManager.java which is basically a tax calculator.

The Model layer is represented by the EmployeeModel.java class. It’s role is to validate the data passed from the presenter and populate its data structures for Employee and PaySlip upon data validation.

The project can be imported in Eclipse as General Java Project. The entry point for the MvpPayRollApp is PayRollView.java. it can be ran by Run as -> Java Application.