|  |
| --- |
| **Hands-on Exercise Objective** |
| After completing the hands-on exercises, you will be able to:   * Develop simple Java program using Wrapper Classes. |
| **Problem Statement 1:**  Develop and simple java application to calculate the school fees for a particular student based on the grade he scored in the entrance examination.  Develop a class named “**Student“** with a method named **calculateFeesStructure** with the following parameters   |  |  | | --- | --- | | Variable Name | Data Type | | studentId | Java.lang.Long | | studentGrade | Java.lang.Character | | monthlyFees | Java.lang.Double | | isScholarshipEligible | Java.lang.Boolean | | Fees | Java.lang.Double |   The business logic for calculating the fees is as follows, this has to be implemented inside the method,  If studentId is not equal to 0 and grade is equal to ‘A’ and isScholarshipEligible is true  fees =monthlyFees -monthlyFees \*10/100 (10% exemption)  Otherwise If studentId is not equal to 0 and grade is equal to ‘B’ and isScholarshipEligible is true  fees =monthlyFees -monthlyFees \* 8/100 (8% exemption)  Otherwise If studentId is not equal to 0 and grade is equal to ‘C’ and isScholarshipEligible is true  fees =monthlyFees -monthlyFees \* 6/100 (6% exemption)  Otherwise If studentId is not equal to 0 and grade is equal to ‘D’ and isScholarshipEligible is true  fees=monthlyFees -monthlyFees \* 4/100 (4% exemption)  Otherwise  Display the message “Not Eligible for Exemption”.  The fees should be displayed as follows,  **“<Exemption %>% of fees is exempted, the calculated fees is <fees> ”**  **IMPORTANT NOTE: The fees printed in the message should be rounded to the nearest int value.**  The test cases to be executed are,   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Test Cases** | **studentId** | **grade** | **monthlyFees** | **isScholarshipEligible** |  | | Test Case 1 | 234 | C | 600 | True | **6 % of fees is exempted, the calculated fees is564** | | Test Case 2 | 115 | B | 909.50 | True | **8 % of fees is exempted, the calculated fees is837.** | | Test Case 3 | 980 | G | 1810 | False | **Not Eligible for Exemption** |   In the Student class add a main method which sets the values and invoke the calculateFeesStructure in the ***Student*** object.  **Problem Statement 2:**  Develop a method named **“compareMarks”** with the following Parameters   |  |  | | --- | --- | | Variable Name | Data Type | | Maths | Java.lang.Long | | English | Java.lang.Double |   The method logic should be as follows,  Convert the English Mark from Double to java.lang.Integer, Maths mark from Long to java.lang.Integer and print the highest mark.  If(English is high)  Display “**English** **mark is higher than Maths**”  If(Maths is high)  Display “**Maths** **mark is higher than English**”  If(equal)  Display “**Both Are Equal**”  **NOTE:**  Use the API of Wrapper objects for comparing and converting into the appropriate wrapper object. Refer the Java doc for the wrapper API’s <http://docs.oracle.com/javase/7/docs/api/>.  **Test Cases:**  Execute the following test cases   |  |  |  |  | | --- | --- | --- | --- | | **Test Cases** | **English** | **Maths** | **Output** | | Test Case 1 | 85 | 65 | English mark is higher than Maths | | Test Case 2 | 56 | 98 | Maths mark is higher than English | | Test Case 3 | 84 | 84 | Both Are Equal | | |
| **Problem Statement 3:**  Develop a method named **“validateFees”** with the following Parameters   |  |  | | --- | --- | | Variable Name | Data Type | | Fees | Java.lang.Double |   The method logic should be as follows,  It should check if the provided number is an infinite or not and display the byte value of the number.  If(fees is infinite)  Display “**Fees is infinite**”  If(fees is **not** infinte)  Display “**Fees is not infinite**”  Then display the **byte value**.  **NOTE:**  Use the API of Wrapper objects for performing the logic. Refer the Java doc for the wrapper API’s <http://docs.oracle.com/javase/7/docs/api/>. | |