Day 3

Java (based on the java day two contents covered: functional programming, run time polymorphism, collections/arrays, String/Date Manipulation)

Problem Statement: This is a progressive / continuation assignment for the previous day (day 2). Replace object oriented programming to functional programming. Add date/time evaluation functionalities where days are added or subtracted to give new date and two dates are added or subtracted to give difference. Create a history which holds all the expressions that are evaluated. Create a user define exception Invalid\_date\_format. At the end of this assignment our calculator should evaluate Arithmetic, Scientific, trigonometric and date functionalities.

Use:

1. Functional programming / lambda expression
2. String tokenizer
3. Date and calendar functions
4. Runtime polymorphism
5. Collections
6. Use Factory design pattern to evaluate either date expression or Arithmetic expression.

Solutions:

Step 1: create new java project 2.

Step 2 : copy the src folder from previous project 1.

Step 3: Create user define exception Invalid\_date\_format Exception.(similar to day 1)

Step 4: delete Arithmetic abstract class and its implementations Add, Sub, Mul, Div, Percetn class. Which is oops now this has to be replaced with functional programming.

Step 5 : create an interface Adaytodate with method public abstract double cal(double a, double b);

Step 6: create an interface Arithmetic with method public abstract double cal(double a, double b);

Step 7: AirthmeticFactory class which to implement the functional programming for Arithmetic operations as shown in demo.

Step 8: Add Date evaluator class to evaluate date as below using functional programming .

* 1. Add (date, integer ) : number of day is added to date to give new date
  2. Add (date, date): two dates are to give new date
  3. Sub(date, integer) number of days are subtracted from existing date to give new date
  4. Sub(date, date) two date are subtracted to give number of day in difference(integer value as output)
  5. Create user define Invalid\_date\_format\_Exception to handle date format error.
  6. Create a DateFactory class which implements runtime polymorphism and return the object of the suitable date operation for the above.

Step 9 Now you should have Date\_Evaluator class to evaluate the date and Expression\_Evaluator to evaluate Arithmetic expressions, scientific expression and trigonometric expression.

Step 10: Create a Factory abstract class to return the object of Date\_Evaluator or Expression Evaluator which has a abstract method evaluator. When input string is given evaluator method is called to evaluate either expression or date. Implement using run time polymorphism.

Step 11 add the below in evaluate abstract class.

1. Add collection to store history data.
2. Add method addHistory to add history of Expression or date.
3. Add method displayHistroy to display all the history. Use functional programming.
4. Use StringTokenizer class if necessary.

Step 12: create main method to check data and arithmetic expression result.