Principles of Object Oriented Programming – Assignment 2

מגישים:
איתי בוגנים, 305278384
סהר ויה, 205583453

Polynomial Calculator

Detailed description of the Classes (fields and methods):

❖ Presentation Package:

CLI Class:

Fields:

- Scanner scan : A scanner to receive user input through command line interface.

Methods:

- void Initiate(): Initial load of the menu and initialization of the user input scanner.
- void menu(): Prints menu and asks for user choice of operation.
- void chooseField():Asks the user to choose over what field will he wish to perform all the calculations and then set up the chosen field in the calculator.
- void addition():Prints instructions for the user about the addition process and then call the calculator addition method to perform the calculations.
- void multiplication():Prints instructions for the user about the multiplication process and then call the calculator multiplication method to perform the calculations.
- void evaluation():Prints instructions for the user about the evaluation process and then call the calculator evaluation method to perform the calculations.
- void derivate():Prints instructions for the user about the derivation process and then call the calculator derivate method to perform the calculations.
- String error(): A function that is called if user input was incorrect or not valid, the method returns a string representing an error with instructions and input expectations.
- void exit():Exits from the application.
- void NewScreen():Clears the screen and prints welcome message to the user.

❖ Coefficients Package:

Scalar <<interface>>:

- Scalar add(Scalar s): Implementation varies, adds two scalar implemented types.
- Scalar mul(Scalar s) : Implementation varies, multiplies two scalar implemented types.
- Scalar pow(Scalar exponent) : Implementation varies, raises a scalar implemented type to the power of an exponent scalar.
- Scalar neg(): Implementation varies, multiplies a scalar implemented type by negative 1.
- Scalar inv(): Implementation varies, inverses a scalar implemented type (1/scalar).
- boolean equals (Scalar s): Implementation varies, returns true if one scalar type is equal to the input scalar type.
- boolean isNegative(): Implementation varies, returns true if the scalar value is a negative number.
- T getValue(): Implementation varies, returns the value of the scalar.
- void setValue(T value) : Implementation varies, sets the value of the scalar.
- Integer getIntegerValue(): Implementation varies, returns the rounded integer value of the scalar.
- void setIntegerValue(Integer value) : Implementation varies, sets integer value of the scalar.
- String toString(boolean isExpZero) : Implementation varies, returns a string representing the scalar type.

RealScalar Class:

Methods:

- Double value : The decimal numeric value of the real scalar.

Methods:

- Scalar add(Scalar scalar): Adds a real scalar to the current scalar.
- Scalar mul(Scalar scalar): Multiplies a real scalar by the current scalar.
- Scalar pow(Scalar power): Raises current real scalar to the power of the input real scalar.
- Scalar neg(): Negates the value of the current real scalar.
- Scalar inv(): Inverses the current real scalar.
- boolean isNegative(): Checks if the current scalar is negative.
- All getters and setters that implements the interface.

RationalScalar Class:

Fields:

- Integer numerator : An integer representation of the numerator part of the fraction.
- Integer denominator : An integer representation of the denominator part of the fraction.
- Double decimalValue : A numeric decimal value of the division (numerator/denominator).
- boolean reduced : Determines whether the fraction reduced already.
- boolean isNatural : Determines whether the fraction represents a natural number (numerator/1).

Methods:

- Scalar add(Scalar scalar): Adds a rational scalar to the current scalar.
- Scalar mul(Scalar scalar): Multiplies a rational scalar by the current scalar.
- Scalar pow(Scalar power): Raises current rational scalar to the power of the input real scalar.
- Scalar neg(): Negates the value of the current rational scalar.
- Scalar inv(): Inverses the current rational scalar.
- boolean isNegative(): Checks if the current scalar is negative.
- void valueToRational(Double value): Receives a double value and convert it to a rational scalar parameters. Sets the numerator and denominator fields according to the conversion.
- void fractionReduction(): Checks if the fraction can be reduce using gcd function. If it can be reduced, reduces the fraction parts.
- int gcd(int a, int b): Find the greatest common divisor between two numbers.
- boolean CheckNatural(): Checks if the current rational scalar is a natural number.
- Integer toNatural(): Convert the rational scalar to natural (denominator 1).
- All getters and setters that implements the interface.

❖ Polynomials Package:

PolyTerm Class:

Fields:

- Scalar coefficient : The coefficient of the polynomial term.
- Scalar exponent : The exponent/power that the variable in the polynomial term is raised by.

Methods:

- boolean canAdd(PolyTerm term): Check if the input PolyTerm can be added to the current PolyTerm by checking if they have the same power.
- PolyTerm add(PolyTerm term): Checks if addition between current PolyTerm and input PolyTerm can be performed.
  Returns a PolyTerm which is the result of the addition if possible, if not returns null.
- PolyTerm mul(PolyTerm term): Performs multiplication between current PolyTerm and input PolyTerm and return a PolyTerm which is the result of the multiplication.
- Scalar evaluate(Scalar scalar): Evaluates the current PolyTerm with an input scalar and returns a scalar which is the result of the evaluation.
- PolyTerm derivate(): Derivates the current PolyTerm and returns a new PolyTerm which is the derivative of the original PolyTerm.
- boolean equals (PolyTerm pt) : Returns true if the current PolyTerm is identical to the input PolyTerm, else false.
- String toString() : Returns a string representation of the polynomial ter
-

Polynomial Class:

Fields:

- List<PolyTerm> terms : A list of all polynomial terms that consists the polynomial expression.

Methods:

- Polynomial add(Polynomial poly): Adds input polynomial to current polynomial and returns a polynomial which is the result of the addition.
- Polynomial mul(Polynomial poly): Multiplies input polynomial and the current polynomial and returns a polynomial which is the result of the Multiplication
- Scalar evaluate(Scalar scalar): Evaluates the current polynomial with an input scalar and returns a scalar which is the result of the evaluation.
- Polynomial derivate(): Derivates the current polynomial and returns a new polynomial which is the derivative of the original polynomial.
- boolean equals (Polynomial poly) : Returns true if the current polynomial is identical to the input polynomial, else false.
- String toString() : Returns a string representation of the polynomial expression.
- int numberOfTerms (): Returns the size of the polynomial that is determined by the amount of polynomial terms that it consists.
- void addTerm (PolyTerm term): Adds a term to the polynomial list of terms.

❖ Logic Package:

StringAnalyzer Class:

Description: A class that is responsible for parsing a user input polynomial string. Will return a valid polynomial expression decomposed to its factors (scalars, coefficients and exponents) in order to perform operations on the given string expression.

Fields:

- Map<String, T> termMap : A map that matches a String described exponent to a Type T coefficient value.

Methods:

- Scalar analyzeToScalar (String scalar): Parses a string type input scalar and returns a Scalar type scalar according to the field the end-user chose to work over.
- Polynomial analyzeToPolynomial (String polynomial): Parses a string type input polynomial, decomposes it to the polynomial terms and returns a Polynomial type expression according to the field the end-user chose to work over.
- Map<String, T> stringToComponents (String poly): Analyzes the input string to its components. Splits the string do separate variable input and differentiate between exponents and coefficients, return a map that matches exponents to the correct coefficients.
- void classifyCoefficients (Map termMap, String coefficient , String exponent): Classifies coefficients to the correct field that is in current use. Adds to the terms map the exponent that match the each coefficients and checks the term map to see if a key of the current exponent already exists. If current exponents already exists in the terms map, adds common power coefficients to a single term.

Calculator Class:

Fields:

- Character fieldRepresentation : A character representation of the current field used by the calculator.
- void Initiate(): A string analyzer used to parse user input strings.

Methods:

- main(String[] args) : Main method.
- void Initiate(): Initialize the calculator (sets current scalar field to null and constructs the string analyzer).
- String addition(String term1, String term2): Parses two polynomial strings to Polynomial expressions using the stringAnalyzer and returns a string representing the polynomial which is the result of the addition of the two input polynomials.
- String multiplication(String term1, String term2): Parses two polynomial strings to Polynomial expressions using the stringAnalyzer and returns a string representing the polynomial which is the result of the multiplication of the two input polynomials.
- String evaluation(String term, String stringScalar): Parses a polynomial string and a scalar string to Polynomial expression and A Scalar type using the stringAnalyzer and returns a string representing the scalar which is the result of the evaluation.
- String derivate(String term): Parses a polynomial string to Polynomial using the stringAnalyzer and returns a string representing the Polynomial which is the result of the derivation of the input polynomial expression.
- Scalar getScalarOverField () : Returns a new Scalar type depending on user choice of field.
- void setFieldRepresentation(Character representation): Sets the current Scalar field character representation determined by user choice

Assignment 2 - Polynomial Calculator
UML Design