EX1 is an assigment that represents three functions; Monom, Polynom and ComplexFunction. This project allows us to make a varrity of math operations towards these functions. In addition we may read a file with a colections of functions and draw them with stddraw in class Functions\_GUI.#class Monom.

**#Class ComplexFunction**

This class implements complex\_function which extands function. This class holds a complex function which is made of an Operation and two functions. The data structure which holds this complex function is a binary tree, the root represents the operation and left and right are represented by a function each. A valid complexFunction has an two arguments for each operation: operation(function,function).

Functions of class ComplexFunctions;

Constructors:

* ComplexFunction()
* ComplexFunction(function f)
* ComplexFunction(String string, function f1, function f2)
* ComplexFunction(Operation operation, function f1, function f2)

plus(function f1)- adds to this complex\_function the f1 function.

mul(function f1)- Multiply this complex\_function with the f1 function.

div(function f1)- Divides this complex\_function with the f1 function.

max(function f1)-Computes the maximum over this complex\_function and the f1 function.

min(function f1)-Computes the minimum over this complex\_function and the f1 function

comp(function f1)-This method wrap the f1 complex\_function with this function: this.f(f1(x))

Function left()- returns the left side of the complex function - this side should always exists (should NOT be null).

Function right()-returns the right side of the complex function – this side might not exist (null)

Operatin getOp()-The complex\_function operation: plus, mul, div, max, min, comp.

chechkWhichOperation(String string)-Checks which operation the string represents and returns the operation (throws exception if operation doesn’t match).

checkWhichString(Operation operation)- Checks which string the operation represents and returns the string (throws exception if operation doesn’t match).

buildComplexFromString-this method builds a complex function from a string recursively (separates from operation f1 and f2). It receives a string which will be divided to operation left and right.

findMainComma(String string)-this method assists the buildComplexFromString function to separate the given string by finding the comma that belongs to its Operation in the beginning of the string if we find a comma and the number of the left brackets are equal to the right brackets then we've found our comma that divides our string.

f(double x)-this method calculates the value of this complex function by a given number recursively.

initFromString(String s)-this method gets a string and converts it to a complex function.

nathOperation(function f1, Operation op)- The method checks which operation it needs to add as root and checks which function it received and puts it as f2 (right) it recevies f1 which may be Monom Polynom or Complex function and an op - operation

toString()-return a String representing this complex function.

quals(Object obj)- Indicates whether some other object is "equal to" this one.

Copy() this function copies this complex function to a new complex function and returns it.

\*\*\*There is a programming problem to check if two complex functions are equivalent for any x between – ∞ and +∞ for obvious reasons therefore I’ve chosen a range with jumps of 0.1 to examine if both functions are equal for all these given points.

**# Class monom** gets a string and checks if it’s a valid monom. Monom has two objects: power (int) which represents the exponent of the monom and the other object is coefficient (double) which holds the valid coefficient of the monom. A valid Monom is one from the shape of aX^b and may have the plus and minus symbols at a beginning of a monom, a - can be a double only, and b - may only b a valid int. A monom can have only the a factor or ax however if it has the b factor it has to come after the ^ symbol.

Functions of class Monom;

Derivative-this method returns the derivative of Monom.

F(double x)-this method returns the value of the monom by a given x(double)-f(x).

isZero- this method returns true if the coefficient is zero.

Monom(String s)- by a given string this method checks if its valid and adds to monom’s parameters its values.

Add(monom m)- this method adds two monoms to one single monom only if they hold the same power.

Subtract(Monom m) this method subtracts two monoms to one single monom only if they hold the same power.

Multiply(Monom m) -this method multiplies two monoms to one single monom.

Equals(Object obj)-this method returns true if this Monom and a given Object are equivalent.

toString() this method prints the monom.

checkString(Stirng s) this method checks if our string represents a valid monom (for example, may have spaces but not characters that aren’t mathematics).

isZero(String s)- this method checks if the string is equivalent to the monom zero.

Parse(String expression)-function that returns its string numeric value with its negative/positive symbol.

initFromString(String s)-this method gets a string and converts it to a Monom.

Function copy() this methos copies this Monom to a new one and returns it

**# Class Polynom**

Class Polynom is a class which constructs Polynom which is represented by a hashmap of monoms and the class has a variety of functions that w can use on the polynom. The class user the help of class monom to make sure we have a valid polynom and for assistant with a few of the functions.

Examples for correct Polynoms: “x”,”x^2 + 3x +4x^43-43”,”43 -x”. (spaces are allowed and +/- symbols). Polynom may not include parenthis in addition to monom’s regulations.

Polynom(string s)- this method adds monoms to our new hashmap and uses class monom to check if the string is a valid monoms.

Double f(double x)-with the help of f function in monom class this method returns the value of the Polynom by a given x.

deleteMonomIfZero(Monom m1)-deletes a monom if its coefficient is zero.

Add(Polynom\_able p1) adds to our Polynom a given polynom\_able.

Add(Monom m1) adds a monom to our polynom & deletes it if its zero.

Subtract(Monom m1) subtracts a monom from the polynom & deletes it if its zero.

Subtract(Polynom\_able p1) subtracts a polynom from the polynom.

Multiply(Polynom\_able p1) adds a polynom to a polynom.

Equals(Object obj) checks if this polynoms and an Object are equal.

isZero() checks if the polynom is zero.

Root(…)-gets two double values and epsilon and returns if there is a value between the two that gives us zero if we send it to f(x) and is smaller then epsilon.

Copy() this method returns a polynom\_able which is a copy of our polynom.

Derivative() returns the polynom with its derivative values.

Area(…)-gets two double values abd epsilon and returns the are of the function between and the x axis.

Iterator()-with this method we may iterate through hashmap or polynom etc. and get its values (monom etc.).

covertMapToList()- this method is used to copy a hashmap to an arraylist. Used for the iterator function to prevent us from trying to acsses data which does not exist (null).

Multiply (Monom m1) this method multiplies our Polynom by one monom.

toString()- this method prints our Polynom.

initFromString(String s)-this method gets a string and converts it to a Polynom.

Function copy() this methos copies this Polynom to a new one and returns it

**#Functions\_GUI class**

The Functions\_GUI class implements the interface functions that represents a collection of functions that we can save from to a file or read from a file and represent them in a graphic window. This class uses StdDraw library that implements all the functions we need to be able to open and draw in the graphic window. We used a LinkedList of type function as a way of linking all our different functions, that we want to add to the graphic window, together.

Functions\_GUI()- initializes the LinkList ‘funds’

add(function e)- received a function and adds is to ‘funcs’ , returns true if the LinkedList changed as a result from the call. returns false if function e already existed and does not add it

addAll(Collection<? extends function> c)- Adds all of the elements in the specified collection to ‘funcs’ returns true if this collection changed as a result of the call

clear()- removes all of the elements from ‘funcs’ contains(Object o)- returns true if and only if ‘funcs’ contains at least one element e such that Objects.equals(o,e).

containsAll(Collection<?> c)- returns true if this collection contains all of the elements in the specified collection

isEmpty()- returns true if ‘funcs’ contains no elements

iterator()- returns an iterator over the elements in ‘funds’

remove(Object o)- removes a single instance of the specified element from ‘funcs’, if it is present

removeAll(Collection<?> c)- removes all of ‘funcs’ elements that are also contained in the specified collection

retainAll(Collection<?> c)- retains only the elements in ‘funcs’ that are contained in the specified collection

size()- Returns the number of elements in this collection

toArray()- returns an array containing all of the elements in ‘funcs’

toArray(T[] a)- adds all elements of ‘funcs’ into an array of type and returns the array

initFromFile(String file)- initializes a new collection of functions from a given son file

saveToFile(String file)- saves a collection of functions to a given json file

drawFunctions(width,height,Range rx, Range ry,resolution)- draws all the functions in the collection in a GUI window using the given parameters for the GUI windo and the range & resolution