Programmation fonctionnelle

Rapport Projet Scala

Bibliothèque de calcul symbolique

Réalisé par :

Jonathan BESSE
Arthur BOUCHERY

Github: projet Scala

Besse Jonathan et Bouchery Arthur

Travail à faire: Bibliothèque de calcul symbolique

Définition des tâches:

Fonctionnalités à implémenter

- fonctions pour convertir une fonction arbitraire en polynôme (lorsque c'est possible) et vice versa
- fonctions pour évaluer un fonction en un point, et de même pour les polynômes (la valeur retournée sera elle aussi de typeRational)
- fonctions pour calculer la dérivé d'une fonction ou d'un polynôme
- fonction pour trouver les racines d'un polynôme (de degré 1 ou 2,voir 3 si vous y parvenez)
- fonction pour calculer la limite d'une fonction en un point (à droite ou à gauche).
  Pour cette fonction, il faudra définir un type qui étende les rationnels avec deux valeurs $+\infty$ et $-\infty$.

Structures de données

- un type de données Rational pour représenter les nombres rationnels,ainsi qu'une classe annexe RationalIsFractional qui implémentera le trait Scala.math.Fractional[Rational], afin de fournir des opé-rations numérique sur le type
- un type de données SymbolicFunction pour représenter des fonctions arbitraires (à une variable, de type Rational) par leur arbre syntaxique
- un type de données Polynomial pour représenter des polynômes (à une variable, de type Rational)

Travail réalisé:

Notre rendu final comporte l'ensemble des structures de données que nous avons tenté de mettre en place tout au long de ce projet.
La structure de données *Rational* est définie comme 2 entiers, appelés ici *a* et *b*.

```
import math.Fractional.Implicits.infixFractionalOps
import math.Integral.Implicits.infixIntegralOps
import math.Numeric.Implicits.infixNumericOps

class Rational(val a: Int, val b: Int) {
  def this(a: Int) = this(a, 1)
  def this() = this(1,1) //Choix arbitraire

  override def toString: String = this.a+"/"+this.b
}
```

La fonction toString servira à l'affichage final pour l'utilisateur.
Les arguments passés en paramètres sont précédés de "*val*" afin de les rendre publics.

La classe *RationalIsFractional* a pour objectif de définir les opérations basiques sur les *Rational* : => division / soustraction / multiplication / addition.
Elle permet aussi de créer un *Rational* a partir d'un *Int*, à partir d'une *String* ainsi qu'une fonction de comparaison.

```
override def plus(x: Rational, y: Rational): Rational =
  new Rational((x.a*y.b)+(x.b+y.a), y.b*x.b)

override def minus(x: Rational, y: Rational): Rational =
  new Rational((x.a*y.b)-(x.b+y.a), y.b*x.b)

override def times(x: Rational, y: Rational): Rational =
  new Rational((x.a*y.a),(x.b*y.b))

override def negate(x: Rational): Rational =
  new Rational(-x.a, x.b)

override def fromInt(x: Int): Rational =
  new Rational(x, 1)

override def parseString(str: String): Option[Rational] =
  val split:Array[String] = str.split( regex = "/")
  if (split(1) == "") None else Some(new Rational(split(0).toInt, split(1).toInt))
```

La classe *SymbolicCalcul* est le type de données qui permet d'effectuer les opérations mathématiques sur les objets *Rational* et *RationalIsFractional*.

La fonction eval permet de définir les opérations basiques sur les *SymbolicFunction* qui prennent comme types de données des *Rational*.
Mais nous avons rencontré des problèmes avec Intellij et nous n'avons pas pu continuer ce projet.

Rétrospective:

Comme vous avez pu le constater nous n'avons pas réussi à faire fonctionner le projet Scala demandé, nous avons passé trop de temps à régler des soucis techniques liés aux versions de Scala incompatibles avec certaines mises à jour de Intellij Idea. Ces soucis techniques nous ont grandement ralenti dans le développement du projet.

Aussi nous avons travaillé ensemble sur les classes qui ont été réalisées (plugin CodeWithMe) sur les mêmes créneaux horaires.

Nous n'avons pas pu résoudre certains des problèmes, le dernier en date étant notre IDE incapable de reconnaître d'autres classes Scala présentes dans le même répertoire.
En résumé, nous avons essayé de faire le projet en suivant une structure similaire à celle des td aussi bien qu'en séparant les classes dans des fichiers différents mais il semble que nous ayons rencontré un problème d'installation.

Conclusion:

Nous avons réalisé les bases du projet avant de rencontrer des problèmes d'ordre technique nous ayant empêché de terminer le projet dans le temps imparti.