

SCALA

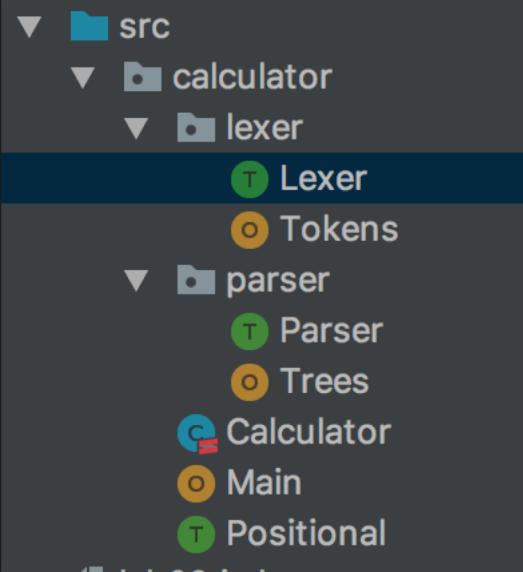
Labo 02 – Calculatrice (again)



Introduction

- Suite de la calculatrice ; gardez les groupes définis dans le labo 1 #dabGuys #pingouins
- Basiquement, on souhaite transformer une chaîne de caractères représentant des opérations mathématiques en arbre syntaxique qui gère les priorités, à l'aide de tokens.
 - Exemple d'arbre pour « 5 + 6 * 2 » : Plus(NumLit(5), Times(NumLit(6), NumLit(2)))
 - Exemple d'arbre pour « (1 + 3^4!) * 5 - 2 % 6 » :
Minus(Times(Plus(NumLit(1), Pow(NumLit(3), Fact(NumLit(4))))), NumLit(5)), Mod(NumLit(2), NumLit(6)))

Arborescence du projet



Lexer: fait le lien entre un symbole (« + », « (», « gcd », valeur numérique, etc.) et un token.
Tokens : contient des tokens (case object) qui représentent un symbole.

Parser : traite les tokens d'une entrée pour construire l'arbre syntaxique (priorité des opérations)
Trees : définit les nœuds et feuilles de l'arbre



Tokens

- Sans paramètre :
 - Symboles mathématiques : '=', '(', ')', '+', '-', '*', '/', '%', '^', '!', ',' (utilisé avec les fonctions)
 - Fonctions mathématiques : "gcd", "sqrt", "invmod" ; insensible à la casse
 - Divers (déjà fournis) : EOF (fin de la saisie), BAD (token erroné)
- Avec paramètre (surcharger la méthode « `toString` » pour garder le paramètre) :
 - Variables pour la mémoire : `ID([String])`
 - Valeurs numériques : `NUM([Double])`

Lexer

- Traitement des nombres : traiter au moins des Int ; bonus si traitement des Double (contrôler que la chaîne de caractères soit bien un nombre à virgule flottant avec au maximum un point, la convertir en Double, ...).
- keywordOrId : traite les strings alphanumériques qui représentent un mot-clé ("gcd", "sqrt", "invmod »), ou une variable pour la mémoire => s'il ne s'agit pas d'un mot-clé, il s'agit donc d'une variable.

Trees & Parser

1. Définir les case class pour chaque nœud dans Trees ; les feuilles sont déjà déclarées.
=> « NumLit » permet de stocker un nombre, « Identifier » stocke le nom d'une variable en mémoire
2. Implémenter le Parser, et tester la sortie de l'arbre sous format String.
=> Par exemple : Minus(Times(Plus(NumLit(1), Pow(NumLit(3), Fact(NumLit(4))))), NumLit(5)), Mod(NumLit(2), NumLit(6)))
3. Implémenter la fonction « compute » dans Trees qui fait le lien entre les nœuds / feuilles et les opérations finales.

Rappels

- Trait, Class, Object, Case Class, Case Object... HELP !
 - Trait : la version Scala d'une classe abstraite en Java
 - Class C : définit une classe « C » qui fonctionne de la même manière que les classes en Java, C++, etc.
 - Class C() { def isSalty(): Int = math.random < 0.9 }
 - val c = new C()
 - c.isSalty()



Rappels

- Trait, Class, Object, Case Class, Case Object... HELP !
- Object O : définit un objet singleton « O » automatiquement instancié à partir d'une classe anonyme, utilisé pour gérer des membres (méthodes, attributs, ...) statiques. L'objet met à disposition les méthodes « apply » (qui agit comme un constructeur) et « unapply » (qui agit comme un déconstructeur => retourne les attributs d'un objet).
 - `Object O() { def isSalty(): Int = math.random < 0.9 }`
`O.isSalty()`
- Case : les classes et objets « case » diffèrent de leurs homonymes ordinaires en offrant un support pour le pattern matching, des implémentations par défaut pour les méthodes « equals », « hashCode » et un joli « `toString` », et pour la sérialisation.