Projet compilation- Grammaire

Prog ::= LOptDef Bloc

LOptDef ::= ε | Def LOptDef

Def ::= **object** IdC **is {** LDeclObjet **}** | **class** IdC (LOptParamClasse) OptExtends **is {** LDeclClasse **}**

LDeclObjet ::= ε | DeclObjet LDeclObjet

DeclObjet ::= **var** Id **:** IdC OptAffectExpr **;** | **def** OptOverride Id **(** LOptParamMethode **)** FinDeclMethode

LDeclClasse ::= ε | DeclClasse LDeclClasse

DeclClasse ::= **var** Id **:** IdC OptAffectExpr **;** | **def** OptOverride Id **(** LOptParamMethode **)** FinDeclMethode | **def** IdC **(** LOptParamClasse **)** OptSuper **is** Bloc

OptAffectExpr ::= ε | := Expr

OptOverride ::= ε | **override**

LOptParamMethode ::= ε | LParamMethode

LParamMethode ::= Id **:** IdC | Id **:** IdC **,** LParamMethode

FinDeclMethode ::= **:** IdC **:=** Expr | OptTypeRetour **is** Bloc

Commenté [LBV1]: Ne pas oublier de faire les variables locales.

OptTypeRetour ::= ε | **:** IdC

Bloc ::= { LInstr } | { LDeclBloc **is** LInstr }

LInstr ::= ε | Instr LInstr

LDeclBloc ::= Id **:** IdC OptAffectExpr **;** | Id **:** IdC OptAffectExpr **;** **LDeclBloc**

LOptParamClasse ::= ε | LParamClasse

LParamClasse ::= OptVar Id **:** IdC | OptVar Id **:** IdC **,** LParamClasse

OptVar ::= ε | **var**

OptExtends ::= ε | **extends** IdC

OptSuper ::= ε | **:** IdC **(** LOptArg **)**

LOptArg ::= ε | LArg

LArg ::= Expr | Expr **,** LArg

Instr ::= Expr **;** | Bloc | **return** OptExpr **;** | Cible := Expr ; | **if** Expr **then** Instr **else** Instr

OptExpr ::= ε | Expr

Cible ::= Id | Expr . Id

Expr ::= Id | Cste | String | (Expr) | (as IdC : Expr) | Selection | new IdC (LOptArg) | Expr . Id (LOptArg) | IdC . Id (LOptArg) | ExprOperator

ExprOperator ::= Expr … Expr (faire avec RELOP,+,-,*,/,CONCAT) | + Expr | - Expr

Selection ::= Expr . Id // A exclure les Expr qui ne marchent pas ici dans la vérification contextuelle

Commenté [LBV2]: Factoriser avec LExpr ? 😊

Commenté [LBV3]: Peut-être une vérification contextuelle 😊 😊 😊 On pourra faire ça après potentiellement 😊 😊 😊