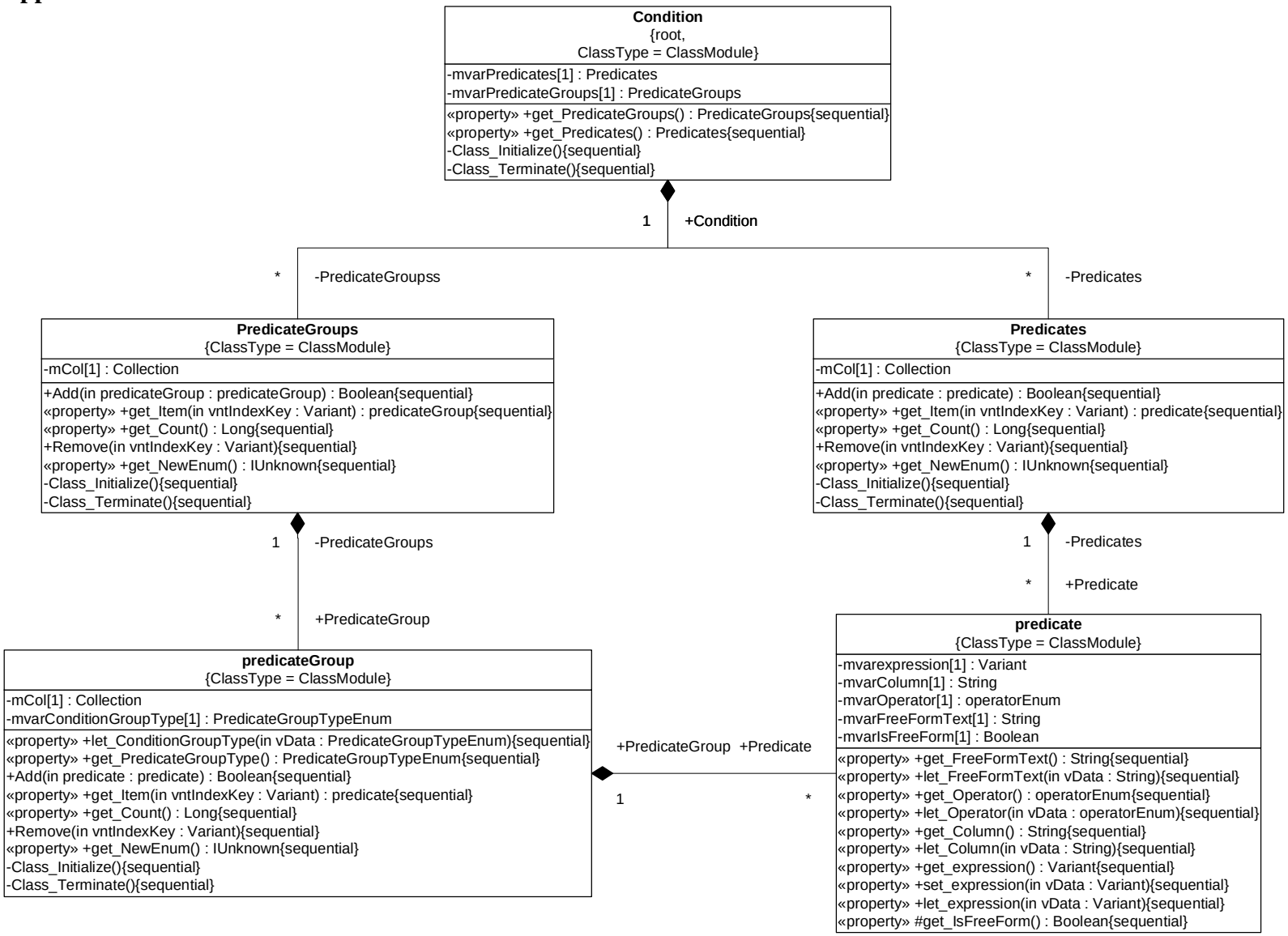
Appendix – I


Condition
{root,
ClassType = ClassModule}
-mvarPredicates[1] : Predicates
-mvarPredicateGroups[1] : PredicateGroups
«property» +get_PredicateGroups() : PredicateGroups{sequential}
«property» +get_Predicates() : Predicates{sequential}
-Class_Initialize(){sequential}
-Class_Terminate(){sequential}
1
+Condition
*
-PredicateGroupss
*
-Predicates
PredicateGroups
{ClassType = ClassModule}
-mCol[1] : Collection
+Add(in predicateGroup : predicateGroup) : Boolean{sequential}
«property» +get_Item(in vntIndexKey : Variant) : predicateGroup{sequential}
«property» +get_Count() : Long{sequential}
+Remove(in vntIndexKey : Variant){sequential}
«property» +get_NewEnum() : IUnknown{sequential}
-Class_Initialize(){sequential}
-Class_Terminate(){sequential}
Predicates
{ClassType = ClassModule}
-mCol[1] : Collection
+Add(in predicate : predicate) : Boolean{sequential}
«property» +get_Item(in vntIndexKey : Variant) : predicate{sequential}
«property» +get_Count() : Long{sequential}
+Remove(in vntIndexKey : Variant){sequential}
«property» +get_NewEnum() : IUnknown{sequential}
-Class_Initialize(){sequential}
-Class_Terminate(){sequential}
1
-PredicateGroups
*
+PredicateGroup
1
-Predicates
*
+Predicate
predicateGroup
{ClassType = ClassModule}
-mCol[1] : Collection
-mvarConditionGroupType[1] : PredicateGroupTypeEnum
«property» +let_ConditionGroupType(in vData : PredicateGroupTypeEnum){sequential}
«property» +get_PredicateGroupType() : PredicateGroupTypeEnum{sequential}
+Add(in predicate : predicate) : Boolean{sequential}
«property» +get_Item(in vntIndexKey : Variant) : predicate{sequential}
«property» +get_Count() : Long{sequential}
+Remove(in vntIndexKey : Variant){sequential}
«property» +get_NewEnum() : IUnknown{sequential}
-Class_Initialize(){sequential}
-Class_Terminate(){sequential}
+PredicateGroup
+Predicate
1
*
predicate
{ClassType = ClassModule}
-mvarexpression[1] : Variant
-mvarColumn[1] : String
-mvarOperator[1] : operatorEnum
-mvarFreeFormText[1] : String
-mvarIsFreeForm[1] : Boolean
«property» +get_FreeFormText() : String{sequential}
«property» +let_FreeFormText(in vData : String){sequential}
«property» +get_Operator() : operatorEnum{sequential}
«property» +let_Operator(in vData : operatorEnum){sequential}
«property» +get_Column() : String{sequential}
«property» +let_Column(in vData : String){sequential}
«property» +get_expression() : Variant{sequential}
«property» +set_expression(in vData : Variant){sequential}
«property» +let_expression(in vData : Variant){sequential}
«property» #get_IsFreeForm() : Boolean{sequential}