Homework3 – Patryk Konieczny – patrykk2

Field(accessType:String, name: String)

The Field method takes an access type as a string and the name of the field which is also a string. This returns a new field object. It is used in the ClassDef method to the create a new field to store in a class definition. This allows the Class Definition of an object to store the necessary data about a field.

Method(accessType:String, name: String, exp: SetOps)

The Method, method takes an access type as a string, the name of the method as a string and an expression that is a setOps expression. This method returns a new method object. It is used in the ClassDef method to store all the relevant method data.

AssignField(name:String, item:Any)

The AssignField method takes a name string and an item of Any. This method returns a map of the name and item. It binds the field to a value. It is used inside the constructor to create that mapping.

Constructor(AssignField(name:String, item:Any))

The Constructor method takes an AssignField Expression. This helps create a binding between a field and a value that is defined in the class. This value is called once a class is instantiated. This implementation only takes one argument of AssignField.

ClassDef(name: String, Field, Constructor , Method)

The ClassDef method creates a new class definition. It takes a name of type string, a Field Method, a Constructor method, and a Method method. This implementation does not consider multiple methods and constructors and nested classes. This Method creates a binding between the name that was given and a new classDef object that holds the field, constructor, and method.

NewObject(name: String, variable : String)

The new NewObject creates a new instance of a class name and creates a binding to the variable. The parameters name and variable are of types string. This returns either the class that is bounded or nothing if it is already bound. This method calls the constructor of the class which created a binding between a field and value. An interface or abstract method cannot be instantiated.

Extends(parentClass: String, childClass: String)

This Extends method creates a link between a parent and child class. The method takes two parameters a parent class name of type string and a child class name of type string. Both classes should be defined before they are extended. The method Maps a child class to the parent to keep records of which class has a parent. An interface cannot extend a concrete class or abstract class. An concrete class cannot extend an interface. A class must implement the abstract methods of an abstract class in order for a concrete class to extend a abstract class.

InvokeMethod(className: String, methodName: String)

The InvokeMethod method executes a chosen method from a chosen class. This method takes a class name as a string and a method name as a string. The class name is the class that you want to invoke the method from, and the method name is the name of that method. In this implementation the class should be instantiated by using the NewObject method prior to using the InvokeMethod method. The class name that is passed into the method should be the variable that is bounded to the class when creating a new instance with NewObject.

AbstractMethod(name)

This method takes a value name that is a string. This creates an instance of an abstract method. This function returns a method object with an empty expression. Evaluated with .evalMethod.

AbstractClassDef(name, Field, Constructor, Method*)

This method defines an abstract class. It takes a name parameter which is a string. It takes a Field class operation, a constructor operation, and Method operations. This returns an AbstractClass object. It can have both abstract and concrete methods but needs at least one abstract method. Evaluated with .evalAbstractClass.

InterfaceDecl(name, Field, AbstractMethod*)

This method declares an interface. It takes a name parameter which is a string, It takes a Field class operation, and AbstractMethod operations. This declares an interface and then return an interface object. Evaluated with .evalClass.

Implements(className, implementationName)

This method takes two parameters of type string. The class name is the name of the class that is implementing the interface that is passed in implementationName. This method returns a string of the interface being implemented. Evaluated with .evalClass