

The Arabic Object-Oriented Programming Language Al-Risalh

Mohammed R. Amin
Department of Electrical and Computer Engineering,
College of Engineering, Umm Al-Qura University,
Makkah, Saudi Arabia.
email: amin@uqu.edu.sa

ABSTRACT

Al-Risalh is an Arabic pure object-oriented programming language that has the basic mechanisms of object-orientation, objects, classes, and messages. Also the language has the full support of object-orientation concepts. The Al-Risalh language supports application programming as well as systems programming. This paper introduces the language that can be used to teach Arabic-speaking students, how to program and how to understand the basic concepts to the idea of object-oriented programming.

Keywords: Arabic object-oriented programming languages, Object-oriented programming concepts, classes and objects, classes and messages.

1. Introduction

So far there does not exist an object-oriented programming language available on a microcomputer or mainframe that can effectively support the development of applications that process Arabic information in a natural way. Computer programming and introductory computer science are still being taught in English due to the lack of an Arabic programming language that can be used to teach the basic concepts of programming. It is very important to communicate the idea of object-oriented programming to Arabic-speaking students since Arabic is the language they understand best.

2. The Language Token

Tokens in Al-Risalh are represented by a sequence of characters that can be treated as a single logical entity. Each token is either an identifier, reserved word, literal, relational operator, delimiter, blank space, or plus-mult-operator.

2.1. Identifiers.

Tokens that start with an Arabic letter are considered to be identifiers. The syntax for an identifier for Al-Risalh is as follows::

```
<identifier> ::= <letter> <identifier-tail>
<identifier-tail> ::= <letter> <identifier-tail> |
<digit> <identifier-tail> |
<stretcher> <identifier-tail> |
_ <identifier-tail> | ε
```

2.2. Delimiters

A computer program is a linear string of characters. To represent the structural elements of this language it is necessary to add delimiters to this string in some way to mark out the blocks, sections, statements, expressions, tokens, and other syntactic elements. Certain symbols are set aside for this process. The delimiters that are used in this language are: () ' - : " ' _ .

2.3. Reserved Words of the Al-Risalh Language

Reserved words are special identifiers that are part of the built-in vocabulary of the language and are used exclusively by the language.

2.4. Literals

The digits are the Arabic numbers 0 through 9 or the Hindi numbers. The syntax for literals is:

```
<number> ::= <digits> <fraction> <exponent>
<fraction> ::= . <digits> | ε
<exponent> ::= E <sign> <digits> | ε
<digits> ::= <sign> <digit> <digits-tail>
<digits-tail> ::= <digit> <digits-tail> | ε
<digit> ::= 0|1|2|3|4|5|6|7|8|9
```

2.5. Arithmetic Operations

In the language, a mathematical expression must be part of either an assignment statement or a logical statement. There are four main arithmetic operators in the language: addition, subtraction, division, and multiplication.

2.6. The Built-in Functions

There are six built-in functions in this language sqrt, abs, sin, cos, sqr, and power.
أس, جذرت, مطلق, جتا, جتا, تربيع, and power.

2.7. Relational Operators

The definition of the Al-Risalh language semantics allows the programmer to assign variables of different atomic types. This will not cause a type mismatch semantic error message. Also the definition of the Al-Risalh language semantics allows the programmer to use the equal operator with class heritage, in which the derived class is assigned to the base class as follows:

```
class circle derived from shape
begin
. . .
end class
message run of myshape
begin
  c : circle;
  s : shape;
  s = c
end message
```

The above assignment statement is legal since every derived class (object) c will be in base class s (object) but not vice versa. In the case of assigning, for example, an array of 10 indices to an array with 20 indices the definition of the language semantics will allow that. To illustrate this, suppose the student declares the follows:

```
list1 = array [1..10]
list2 = array [1..20]
```

and later in the program he assigned list1 to list2:

```
list2 = list1;
```

the above statement is legal since the values in list1 will override the values in list2. Therefore, the values of list2 will disappear and will be replaced by the values in list1. Also the programmer can replace the value of index 7 in list2 by the value of index 3 in list1 as follows:

```
list2[7] = list1[3];
```

the above statement will make the value in index 7 of list2 disappear and it will be replaced by the value in index 3 of list1.

3. Statements

Statements describe algorithmic actions that can be executed. There are seven types of statements.

3.1 Declarative Statements

A declarative statement is a statement that declares one or more variables and specifies their type(s). A variable type can be categorized as either a class type or an atomic type. A variable of atomic types are predefined in the language and include integer, boolean, string, single, double, or char type. The syntax for a declarative statement follows:

<تصريح - جملة> ::= <تعريف> : <نوع - المتغير>;

<decl-statement> ::= <identifier> : <data-type>;

3.2. Assignment Statements

Assignment statements replace the current value of a variable with a new value specified by an expression. An example of an assignment statement follows:

القوة = الحركة + الدفع;
c = a + b;

3.2 Send Statements

Send statements specify the activation of the message denoted by the message identifier. Class messages are invoked only with a send statement. An example of a send statement follows:

أرسل طول_الضلع الى ص مطلوب العدد;
send Edges to x result in count;

3.3 Statement List.

The language semantics specifies that its compound statements are to be executed in the same sequence as they are written. The compound statement has the form:

<جملة>;
<جملة>
<statement>;
<statement>

3.4 Repetitive Statements

The repetitive statement in this language is the **while** statement. The syntax is as follows:

< جملة - تكرارية > ::= < طالما > مقدار - جبري <
 بداية
 < جملة - متتابعة >
 نهاية طالما

<while-statement> ::= **while** <expression>
begin
 <statement-list>
end while

3.5 Conditional Statements

A conditional statement selects for execution a single one (or none) of its compound statements. The syntax for the **if** statement follows:

< الجملة - الشرطية > ::= إذا < مقدار - جبري >
 بداية < جملة - متتابعة > نهاية إذا
 [وإلا بداية < جملة - متتابعة >] نهاية
 [وإلا]

<if-statement> ::= **if** <expression>
begin <statement-list> **end if**
[else begin <statement-list> **end else]**

4. Class Declarations

A class can be a standalone class or derived from a base class. However, this technique resolves the classes and class libraries issue in object-oriented programming. The class syntax is as follows:

< تصريح - نوعية > ::= نوعية < التصاريح - الأولوية - للنوعية >
 بداية
 نوعية مساعدة <
 الرسائل - العامة - للنوعية <
 خصوصيات - النوعية <
 نهاية نوعية

<Class-decl> ::= **Class** <class-decl-head>
begin
 <class-friends>
 <class-public-msgs>
 <class-privates>
end class

The above syntax allows us to have a class as an abstract data type (ADT). And objects of this class are available at run time. The syntax expresses the reusability (**inheritance**) feature; one can derive a class from another class. Also this type of declaration will

help the student when compiling Al-Risalh programs, since Al-Risalh uses **one pass compilation**.

4.1. Class Friend Declarations

A class declared as a friend will have access privileges to the private information. The class friend declaration is as follows:

< نوعية - مساعدة > ::= مساعدة: < تعاريف - اسم - نوعية >

<class-friend> ::= **friend:** <class-name-identifier>

4.2. Class Public Message Declarations

Public messages can be declared as static or **polymorphic** messages. A static message, which makes use of early binding, is executed almost exactly like a normal procedure or function. Polymorphic messages, on the other hand, act very differently. When you call a **polymorphic** message the actual call is bound at run time, when the call is made. This is called late binding. The syntax for public messages is as follows:

< الرسائل - العامة - للنوعية > ::= رسائل عامة: < تعريف - اسم - الرسالة >
 مرفق (< المعطيات - الرسمية - للرسالة >)
 مطلوب < تعريف > : < نوع - المتغير >
 [: متعددة] [<

<class-public-msg> ::= **public message:**
 <msg-name-identifier>
 [<msg-with-clause>]
 <msg-result-clause>][:polymorphic]

4.3. Private Message Declarations

Private messages are private to the class; no other class will have access privileges to them except a friend class. Private messages can be declared static or polymorphic messages. Declaring a message as private will impose the feature of **encapsulation**. The private message syntax is as follows:

< الرسائل - الخاصة للنوعية > ::= رسائل خاصة: < تعريف - اسم - الرسالة >
 مرفق (< المعطيات - الرسمية - للرسالة >)
 مطلوب < تعريف > : < نوع - المتغير >
 [: متعددة] [<

<class-public-msg> ::= **private message:**
 <msg-name-identifier>
 [<msg-with-clause>]

[<msg-result-clause>]
[: polymorphic]

4.4. Private Attribute Declarations

A variable that has been declared in this part will belong to one of two major types, an atomic type, or a class type. An atomic type is a predefined type in this language a class type is a class defined by the programmer. Therefore, you can declare a variable "mysquare" as follows:

دائرة : شكل؛

mysquare : shape;

5. Message Definitions

Class messages are invoked only with a send statement. The message may invoke a class message to perform calculations, access and change the class' instance variables, or send messages to other classes as a part of its response. A class' instance variables can be manipulated only by its own messages. That is, messages telegraph a request for actions to a class while the class' messages decide how to perform that action. The message definition syntax is as follows:

<تعريف - رسالة > ::= رسالة < تعريف - أسم - الرسالة >
< نوعيتها > تعريف - أسم - نوعية <
[< فقرة - ما بطي - الرسالة >]
[< فقرة - ناتج - الرسالة >]
بداية < جمل - متتابعة > نهاية رسالة

<message-defn> ::= message <msg-name-id>
of <class-name-id>
[<msg-with-clause>]
[<msg-result-clause>]
begin <statement-list> end message

A message can also be a recursive message. The following example illustrates this:

رسالة دارة نوعيتها حاسبة مرفق (س : صحيح)
بداية
إذا س < 0
بداية
س = س - 1
أرسل دارة الى نفسي مرفق (س)
نهاية إذا
نهاية رسالة

```
message rec_mes of calc with (x : integer)
begin
  if (x > 0)
    begin
```

```
x = x - 1;
send rec_mes to self with(x)
end if
end message
```

6. Al-Risalh's Features

Al-Risalh is a self-documenting and natural language and that form readability. As of reusability Al-Risalh language has been designed to support reuse, in whole or in part, for new applications. The characteristics of the class declarations syntax impose the inheritance feature. Understandability is applied in its simplified syntax. Extendibility is expressed by applying the polymorphism feature to public and private message declarations. Al-Risalh languages have been designed in a systematic manner to have orthogonal features.

Conclusion

In this paper we have discussed some of the languages features that we feel are important. Al-Risalh is an Arabic object-oriented programming language that has the basic mechanisms of object-orientation, objects, classes, and messages. Also the language has the full support of object-orientation concepts. The Al-Risalh language supports application programming as well as systems programming.

References

- [1] B. Meyer, "Object-Oriented Software Construction", Prentice Hall, Englewood Cliffs, NJ, 1988
- [2] E.M. Arif, "Introduction to Object-Oriented Programming with C++". Jeddah House Publishing, Jeddah, Saudi Arabia, 2001.
- [3] E.M. Arif, "Design of An Arabic Object-Oriented Programming Language and A Help System for Pedagogical Purposes. Ph.D. thesis, Illinois Institute of Technology, Chicago, IL, 1995.