Smalltalk Classes

Background

Smalltalk is an object-oriented programming language that was developed by the Learning Research Group of Xerox PARC in the 1970s. The current standard version of Smalltalk (ANSI Smalltalk) was approved in 1998, and is representative of the most commonly used Smalltalk language. Smalltalk, like most object-oriented languages, focuses on making use of objects to create appropriate classes for an application. The Smalltalk standard can be downloaded for free at http://www.smalltalk.org/versions. Specifically, the ANSI Smalltalk and GNU Smalltalk are available for download and testing of Smalltalk classes and applications.

Getting Started with Smalltalk

Starting Smalltalk is relatively easy once it is installed on your computing system. We will use the GNU Smalltalk as a means of illustrating how to work through some of the concepts that will be explored. From the command prompt, you can launch Smalltalk with the following command:

$ gst

Once all of the appropriate system tools are loaded and the Smalltalk startup prompt is output on your screen you’re ready to start working on a few examples. The appropriate Smalltalk prompt will take on the following appearance on your screen:

GNU Smalltalk ready

st>

A simple “hello world” program can be executed in Smalltalk by using the following command:

'Hello not so small world' printNl !

Class Structure and Syntax in Smalltalk

Smalltalk can be differentiated from other programming languages that contain object-oriented concepts by one main point: all of the components of Smalltalk applications are objects. For example, in C++, C#, or Java the program class can be composed of various primitive data types as well as objects, which affect the structure of the program. However, in Smalltalk all values that are passed are representative of objects in the program. Smalltalk objects that are used in applications are able to provide classes with the capability to carry about three main actions, which include:

I. Holding a reference state for other objects.
II. Acceptance of a message from another object or itself.
III. Transmission and processing of a message to another object or itself.

Since Smalltalk makes use of objects for all components in applications, it is important to note that the structure of applications occur in a tree-like hierarchy. This can be represented as a class composing the roots of the tree. The components that would branch below the class, children nodes, would include data types, such as integers, arrays, and strings.

The overall structure of applications and classes in Smalltalk make use of a small subset of appropriate reserved words or keywords. These keywords are used to determine how variables and identifiers are able to interact with one another within the application. The keywords that are most often used in a Smalltalk application involve: thisContext, super, self, true, false, and nil.

Creating a Smalltalk Class

Smalltalk classes are created by sending a message to create the class. The message is sent to the appropriate object, which denotes the class name that will be used. The following example provides an illustration of how to bind an appropriate class called Food.

Object subclass: #Food
    instanceVariableNames: 'FoodType'
    classVariableNames:"
    poolDictionaries:"
    category: nil !

In this example, we have explicitly formed the object class to add an appropriate subclass called Food. We also declare an appropriate variable using “instanceVariableNames: ‘FoodType’” to denote that each object will contain this variable.

Sources