GOLDBERG83

Goldberg, A., Robson,D.;
SMALLTALK-80 The Language and Its Implementation;
Adison-Wesley series in Computer Science (May 1983);
Part 2 (Ch. 6 - Ch. 20).

In the second part of this four part book, each major component (class and its subclasses) in the system's library (called the virtual image) is described in some detail. The hierarchy of these classes is also presented to give a global view of the system.

Following these chapters, one can see how the virtual image is built up from the basic concepts described in the first part of the book. In doing so, one of the design philosphy of the Smalltalk-80 system (to reduce the amount of effort needed in building large complex systems by making use of previous work) is very well demonstrated. The presentation is generally clear and examples are used to illustrate the usage of each feature although some confusions arise. Actually one chapter is devoted entirely for illustrating small to medium size examples.

Although the virtual image is described in the object (and class) view, in some cases one can interpret it in a more traditional way: consider it an encapsulation of data type and operations defined for data belonging to that type. (It is interesting and constructive to note the differences and similarities between what class/object offer and what abstract data type offer.)

There are many different types of classes supported in the virtual image. Some of these classes support operations on scalar data type (in traditional program-ming language sense), e.g. Float, Fraction and Integer. Other classes support operations on non-scalar tyupe known as collection which is more powerful than array in other programming languages. Collection is a big generic class with many subclasses, with each subclass having different properties; e.g. the Bag subclass allows duplication of elements while the Set subclass doesn't, some subclasses will maintain an ordering of element either internally (e.g. SortCollection) or externally (e.g. LinkedList). Another interesting thing is that array (a subclass of collection) has a subclass which enforce strict typing (e.g. ByteArray allows its element to be SmallInteger only) and a subclass which has dynamic typing (e.g. Array allows its element to be anything).

Smalltalk also has the notion of a process, which can be created by sending the unary message fork or newProcess to a block (a sequence of actions described by expressions enclosed by square bracket "[]"). The object Processor (the only instance of the class ProcessorScheduler) coordinate the execution of all processes ready to go. Processes can be put in foreground and background with different priorities. Running processes can be suspended and resumed later, an active process can be terminated or forced to give way to other process at the same priority. Coordination between running processes (for mutual exclusion access or resource sharing) is possible by the usage of instance of the class Semaphore. (Thus some kind of job control, normally provided by operating system's front end, is embedded in the language itself.)

The last four chapters in this part is devoted to user interface and the system's features (e.g. the graphical kernel) that support it. This graphical -oriented user interface support a high degree of dynamic interaction between the user and the system.

As a conclusion: the understanding of this part, to some extent, is a prerequisite for people who want to create applications using Smalltalk.