

[0030] 3. a static object; and

[0031] 4. a dynamic object.

[0032] An object can simultaneously be more than one of these four types of objects. However, an object can be either a static object or a dynamic object, but not both. As will become clear from the discussion below, generally, an object will either be static or dynamic. Furthermore, an object can be either an object assigned to another object or an object defined within another object but not both.

[0033] Referring now to **FIG. 1**, symbol 1 is used to represent an application, project, or system type object (hereinafter application). These three terms are intended to define the same thing and are not intended to be three different things. Within the relevant professional fields, these terms are generally used interchangeably to refer to an overall computer program. This symbol may be used to represent the most general objects in a computer program. For instance, it may be used to represent the overall program. It also may be used to represent windows, reports, menus, classes, and methods. An application object 1 is a dynamic object and commonly will be neither defined within another object or assigned to another object.

[0034] Symbol 2 is used to represent a window or form type object such as a window in a graphical user interface. (Note that, in the JAVA programming language, a window is called a form.) In accordance with the terminology of the present application, a window is a user interface that contains a set of objects that can execute rules. A window generally is in the top level of the user interface of an application. A window object generally is a dynamic object because, at a minimum, it will have another object defined within or assigned to it. For example, even a window that comprises nothing but a single graphic file still would be a dynamic object in accordance with the terminology as defined hereinabove. The graphic file, on the other hand, would be a static object as it has no logic, rule, event, script or other object assigned to it.

[0035] Merely as an example, in a business application for an e-commerce retailers website, one class object may be called "customer" wherein its attributes, methods and functions define what information can be entered about a customer of the retailer (e.g., name, address, telephone number and what can be done with respect to a customer object, e.g., it can be added, deleted, or modified). A class object generally will be a dynamic object. However, a class may have only attributes and no methods, in which case it would be static. A class generally will not be defined within another object or assigned to another object.

[0036] Symbol 3 represents a menu-type object. It may be used to represent a menu bar, a menu, an item in a menu, or a menu option depending on the desired level of detail for the diagram. A menu is a list of options and commands from which a user can choose one option or command. Again, anyone familiar with graphical user interface programming languages such as Java and HTML is familiar with the concept of menus, menu bars (a collection of menus) and menu items or options. A menu-type object almost always is a dynamic object and is an object assigned to another object. For instance, an object menu does not depend on any other object to exist, but in order for a menu to be useful in most cases, it must be assigned to an object window.

[0037] Symbol 4 represents function/procedure/method type objects (hereinafter "methods"). A method is a well known concept of object oriented programming languages and, therefore, does not warrant a detailed discussion herein. As will be discussed in greater detail below, a method symbol in an OED of the present invention does not represent a call to a method. Instead, it represents the fact that the method is available to the object to which it is assigned. A method object is a dynamic object. Further, it is an object assigned to another object. Particularly, it is not an object defined within another object as it does not require another object for its definition, but generally will be of no use unless it is assigned to another object, such as an application (in which case it can be available to any other object in the application) or a window or a menu.

[0038] Symbol 5 represents an event script type object. An event script is code that is executed when an event associated with an object (e.g., clicking on a button) occurs that causes another piece of code to be run. Particularly, when the event occurs, the event script is executed, which, in turn, causes another module of code to run. Therefore, an event script object is a dynamic object as well as an object assigned to another object.

[0039] Symbol 6 represents a frame or report type object (hereinafter "a frame object"). A frame object is an interface used to display the results of a query, a formatted report, or a set of fields used to accept input from a user. A frame is usually defined within an object window and, therefore, usually is an object defined within another object. Also, it is a dynamic object. With brief reference to **FIG. 2**, which shows a web page, the rectangular box within which the information regarding various flight options is shown is a frame. Particularly, it contains data which is retrieved from a database as a result of a query. For example, in **FIG. 2**, the query was to identify available flights from Rio de Janeiro to New York City. Frames can be relatively complex objects. Formatted data reports are presented within frames. The user interfacing with the program can select data items within a frame and such selections can lead to actions being taken (e.g., an event script or method being executed).

[0040] Symbol 7 represents a class type object. A "class" is a well known element of object oriented programming and has previously been discussed herein and, therefore, will not be described in detail. Briefly, it is a category of objects. A class object defines a set of attributes, functions and methods for objects of that class. It is a general definition of objects of that class. As noted in the background section, a class defines its elements, e.g., attributes, functions and methods, as variables and an actual object is a particular instance of a class.

[0041] Symbol 8 represents a button type object. It represents a button. Once again, button objects are well known to anyone familiar with graphical user interfaces. Types of buttons include option buttons, check buttons, and picture buttons. A button is a dynamic object as it usually will have an event script attached to it (i.e., something happens when you click the button). Further, a button generally will be an object defined within another object. Particularly, a button generally will be of no use unless it is in a window.

[0042] Symbol 9 represents a data structure or record type object. Again, the concept of a structure or record is well known. It is a standard feature of the C++ programming