

## METHOD AND APPARATUS FOR DOCUMENTING AND DESCRIBING OBJECT ORIENTED PROGRAMMING LOGIC

### FIELD OF THE INVENTION

[0001] The invention pertains to the documentation and description of object oriented programming logic, particularly in connection with fourth and fifth generation object oriented programming languages.

### BACKGROUND OF THE INVENTION

[0002] Traditionally, a computer program was viewed as a logical procedure that takes input data, processes it, and produces output data. The process of developing a software routine was seen as a process of determining how to write the logic to achieve the desired actions. Object oriented programming (OOP) was a revolutionary concept that changed the paradigm for computer software development by taking the view that computer programming should be organized around objects rather than actions, i.e., data rather than logic. Today, most software development is performed in object oriented programming languages (OOPs). C++ and Java are among the most popular object oriented programming languages today. The Java programming language is designed especially for use in distributed applications on corporate networks and the Internet.

[0003] In object oriented programming, an object is any data structure with a defined intent that is capable of executing a logical sequence of commands. An object can represent virtually anything, such as a person (e.g., described by name, address and/or other information) or a room (the properties of which can be described by attributes, such as its length, width, furnishings, wall color, floor type, etc.) or elements of a graphical user interface (GUI) such as buttons, scroll bars, pull down menus, windows, etc.

[0004] In object oriented programming, the first step is to identify all objects that one may wish to manipulate and how they relate to each other. This process is known as data modeling. The actual data that defines a particular object such as the aforementioned name, address, length, width, color, furnishings, etc., are called its attributes. An attribute essentially is a changeable property or characteristic of an object that can be set to different values. Once an object has been identified, it is generalized as a "class" of object. The class defines the kind of data (e.g., attributes) that objects of that class contain and any logic sequences, e.g., methods, that can manipulate it. Accordingly, a class basically is a template definition of the methods and variables/attributes of a particular type of object. Thus, an object is a specific instance of a class, i.e., it contains actual attribute values instead of variables. Each distinct logic sequence is known as a method. A method is a programmed procedure that is defined as part of a class and included in any object of that class. A class (and thus an object) can have more than one method. A method in an object can only have access to the data known to that object, which ensures data integrity among the set of objects in an application. A method can be re-used in multiple objects.

[0005] The object (sometimes called a class instance) is what is run on a computer. Its methods provide computer instructions and the class object characteristics provide relevant data. A human operator communicates with

objects—and they communicate with each other—via well-defined interfaces called messages.

[0006] In object oriented programming, an event is anything that occurs that causes a piece of code to be executed. Accordingly, an event can be a human user input event, such as clicking on a button or a hyperlink, or program driven, such as when one piece of code performs some function that results in another piece of code being executed.

[0007] The concept of a data class makes it possible to define subclasses of data objects that share some or all of the main class characteristics. Called inheritance, this property of OOP forces a more thorough data analysis, reduces development time, and insures more accurate coding.

[0008] Software programs can be represented at many levels of detail, such as object code, source code, and higher levels of representation. Object code (or machine code) essentially is numerical data that the actual processor understands. Object code is what is known as a first-generation programming language, or 1GL. Assembly language (example instruction: add 12, 8) is called second-generation language or 2GL. 3GL or third-generation language is a high level programming language, such as PL/I, C, or Java and may or may not be an OOP. A compiler converts the statements of the specific high-level programming language into machine language. A 4GL or fourth-generation language is designed to be closer to natural language than a 3GL language. 5GL or fifth-generation language is programming that uses a visual or graphical development interface to create source language that is usually compiled with a 3GL or 4GL language compiler.

[0009] Some companies, including International Business Machines Corporation, make 5GL visual programming products for developing applications in Java, for example. Visual programming allows a software developer to easily envision object oriented programming class hierarchies and drag and drop icons to assemble program components.

[0010] In the field of software development, a software developer, systems analyst, or program architect (hereinafter "architect") typically prepares a specification describing at a higher level than the actual code level the software that is to be developed and provides that information to a programmer (or coder) who writes the actual software in source code. In traditional (i.e., non-OOP) programming, a software architect may have described the desired software in terms of a flow chart which comprises a series of steps (i.e., actions) in some sequence. Flow charts, however, are not particularly well-adapted to representing object oriented programming because, as previously mentioned, object oriented programming focuses on objects rather than actions.

[0011] Software tools particularly adapted for object oriented programming languages are available today that are designed to allow a software architect to describe the software at a high level so that the software architect can work more efficiently by concentrating on the business or other logic without getting bogged down in the actual computer science of writing code.

[0012] However, the available methods and tools are not as efficient as could be in terms of assisting a software architect in defining, representing, and documenting OOP applications, particularly applications developed to work in a graphical user interface.