**ABSTRACT**

This paper is scrutinizes the use of different concepts of Object Oriented Programming, enabling viewer to get the complete concept of different aspects of Object Oriented Programming. OOP languages allows you to break down your software into bite-sized problems that you then can solve — one object at a time. To satisfy this we created a simple menu-driven program displaying various Hospital Management System. With use of Inheritance and Polymorphism is also used as a reference to the output, satisfying every need of a perfect OOP program.

**Table of Content**

1. Introduction……………………………………………………………………3
2. History…………………………………………………………………………4
3. Design / Implementation……………………………………………………....5-9
4. Flowchart…………………………………………………………..5
5. Code………………………………………………………………..6
6. Output And Analysis…………………………………………………………..10-15
7. Output……………………………………………………………...10
8. Analysis……………………………………………………………15
9. Conclusion And Future Enhancement…………………………………………16
10. Conclusion…………………………………………………………16
11. Future Enhancement……………………………………………….16
12. References……………………………………………………………………...17

**Introduction**

**1. Object Oriented Programming:**

OOP stands for Object-Oriented Programming. Procedural programming is about writing procedures or functions that perform operations on the data, while object-oriented programming is about creating objects that contain both data and functions. Object-oriented programming has several advantages over procedural programming. OOP is faster and easier to execute. OOP provides a clear structure for the programs. OOP helps to keep the C++ code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug. OOP makes it possible to create full reusable applications with less code and shorter development time.

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which can contain data and code: data in the form of fields (often known as attributes or properties), and code, in the form of procedures (often known as methods). A feature of objects is that an object's own procedures can access and often modify the data fields of itself (objects have a notion of this or self). In OOP, computer programs are designed by making them out of objects that interact with one another. OOP languages are diverse, but the most popular ones are class-based, meaning that objects are instances of classes, which also determine their types.

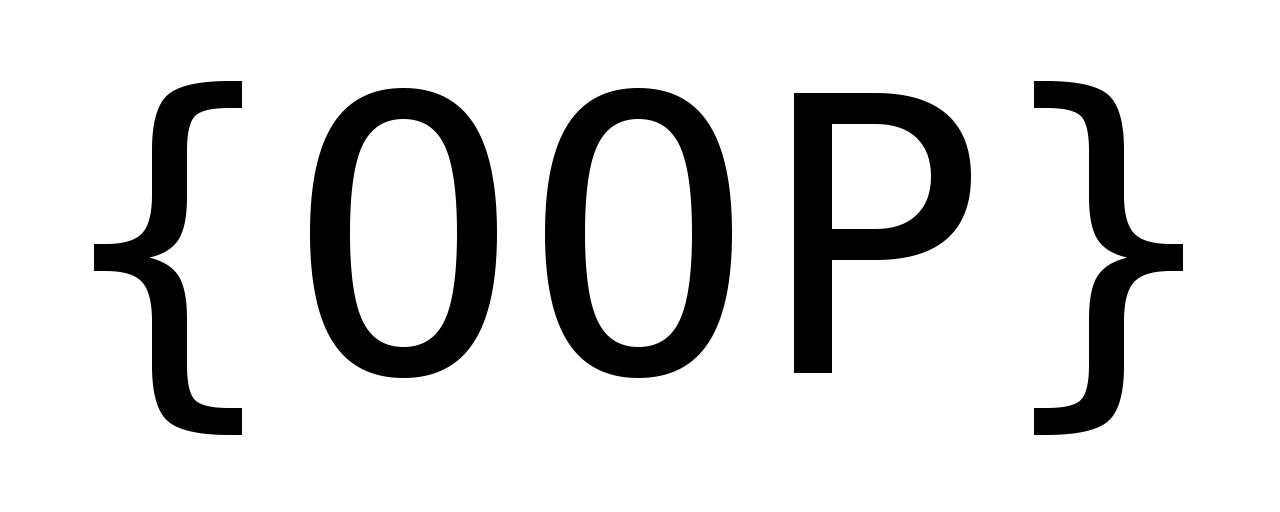


Fig. Object Oriented Programming

**History**

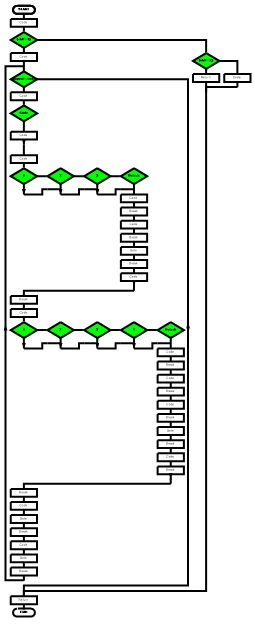
Terminology invoking "objects" and "oriented" in the modern sense of object-oriented programming made its first appearance at MIT in the late 1950s and early 1960s. In the environment of the artificial intelligence group, as early as 1960, "object" could refer to identified items (LISP atoms) with properties (attributes) Alan Kay was later to cite a detailed understanding of LISP internals as a strong influence on his thinking in 1966.He thought of objects being like biological cells and/or individual computers on a network, only able to communicate with messages (so messaging came at the very beginning – it took a while to see how to do messaging in a programming language efficiently enough to be useful).Another early MIT example was Sketchpad created by Ivan Sutherland in 1960–61; In 1962, Kristen Nygaard initiated a project for a simulation language at the Norwegian Computing Center, based on his previous use of the Monte Carlo simulation and his work to conceptualise real-world systems. Ole-Johan Dahl formally joined the project and the Simula programming language was designed to run on the Universal Automatic Computer (UNIVAC) 1107. Simula introduced important concepts that are today an essential part of object-oriented programming, such as class and object, inheritance, and dynamic binding But although the idea of data objects had already been established by 1965, data encapsulation through levels of scope for variables, such as private (-) and public (+), were not implemented in Simula because it would have required the accessing procedures to be also hidden.



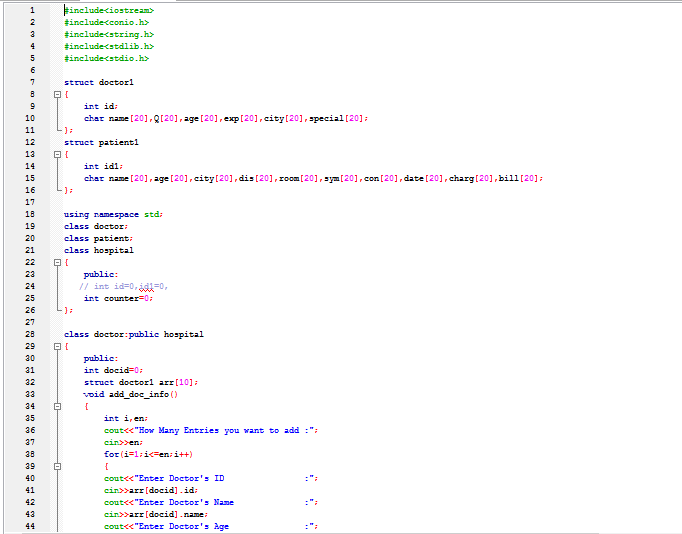
Fig. Alan Kay

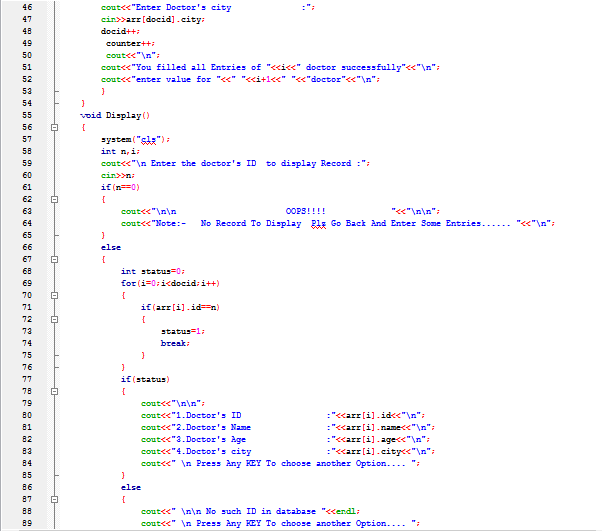
**Design/Implementation**

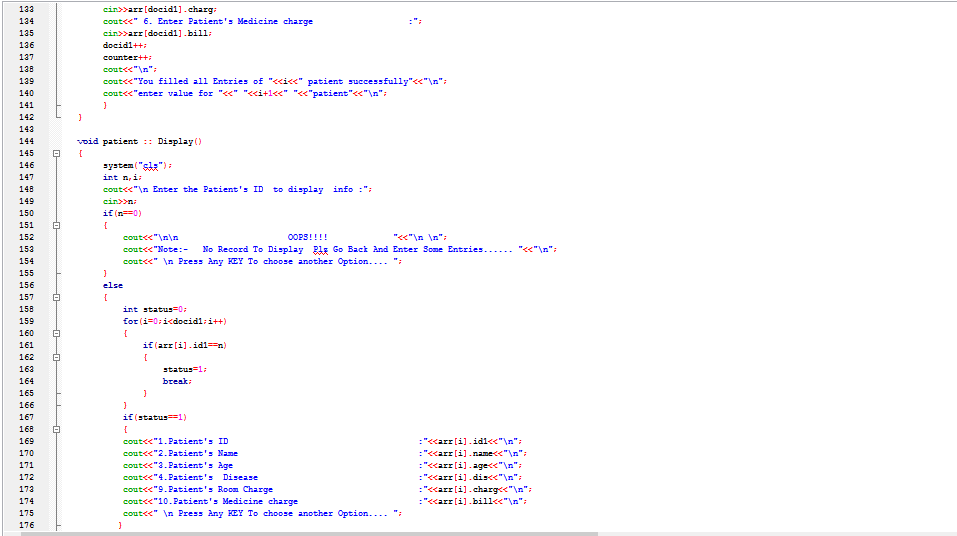
1. **Flowchart**

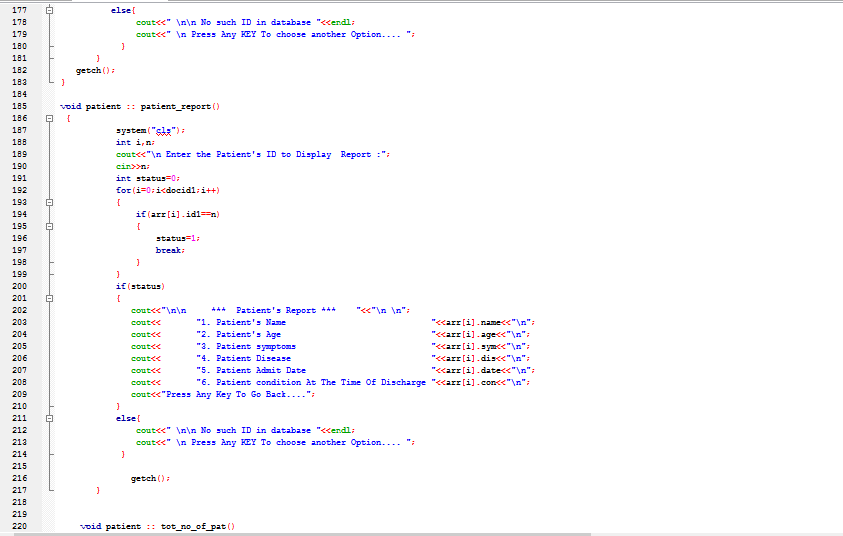
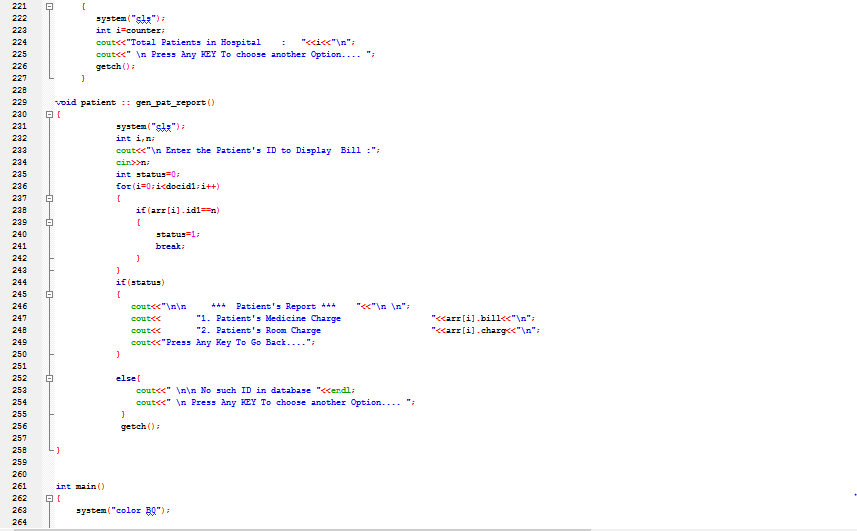
****

1. **Code:**

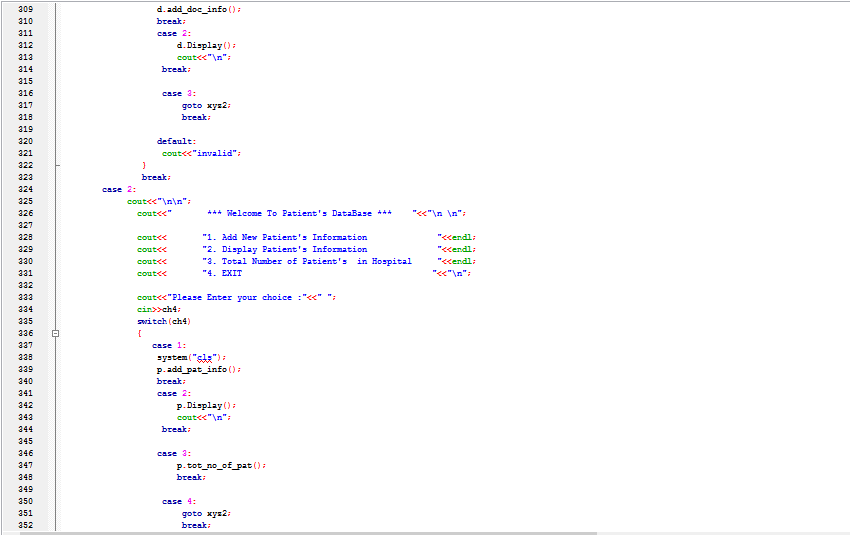
****

****

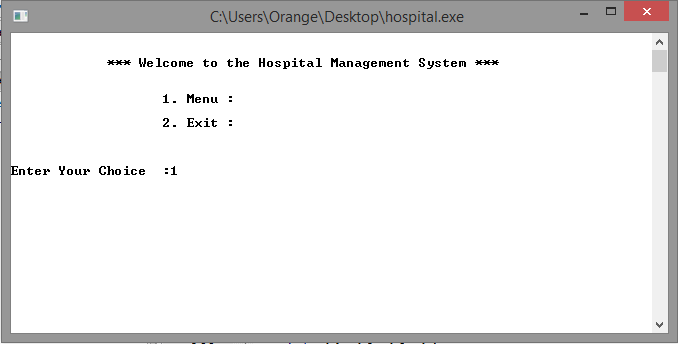
****

****

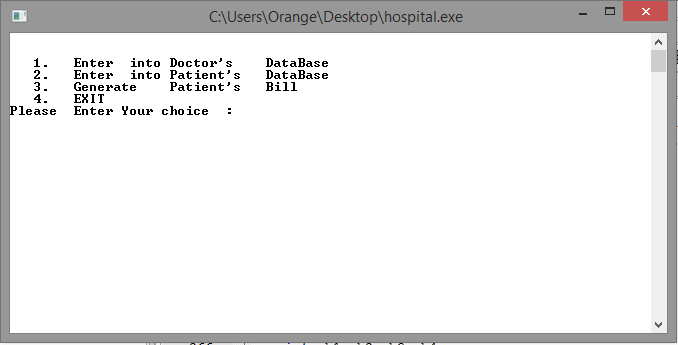




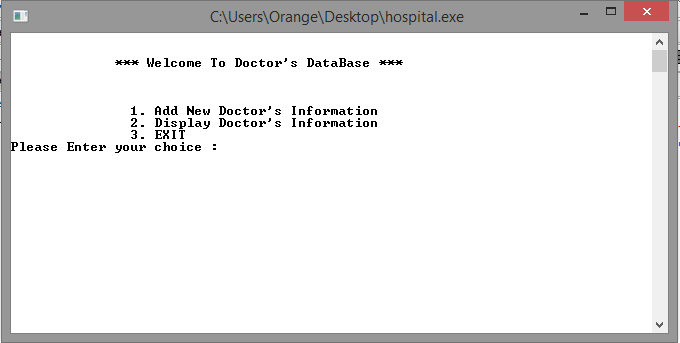
**Output/Analysis**

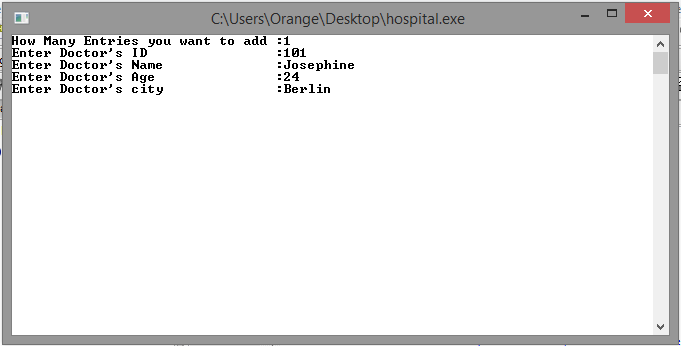
1. **Output**

**Fig. 1. Main Menu**

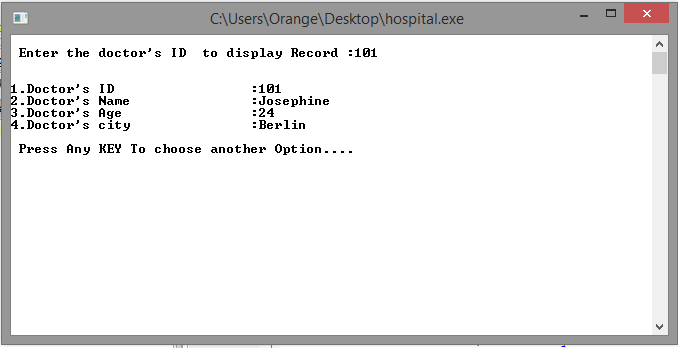
****

**Fig. 2. Menu**

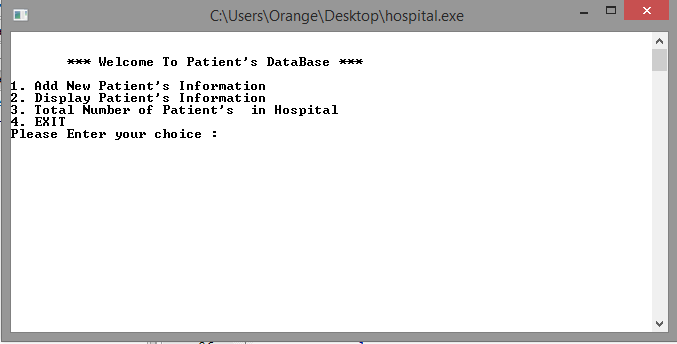
**Fig. 3. Doctor’s Database**

****

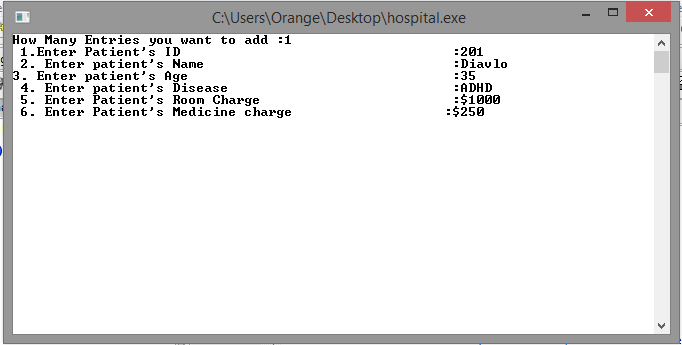
**Fig .4. Add Doctor Information**



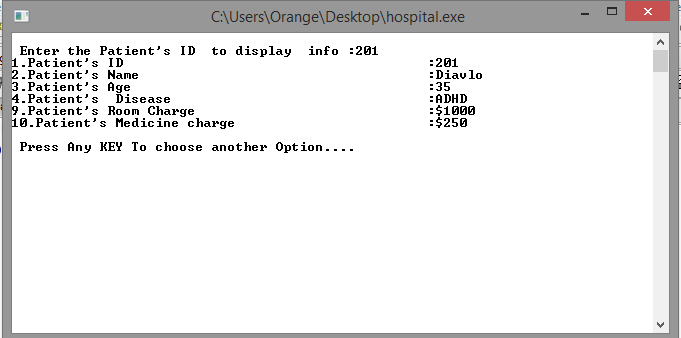
**Fig. 5. Display Doctor Detail**

****

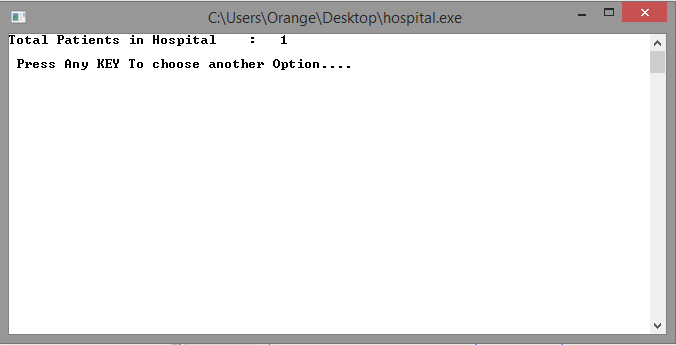
**Fig. 6. Patient Database**

****

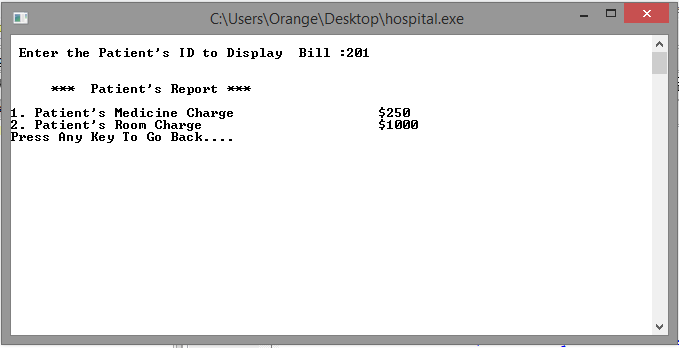
**Fig. 6. Add New Patient**

****

**Fig. 7. Display Patient**

****

**Fig. 8. Total Patients**

****

**Fig. 9. Patient Bill**

**Analysis**

Many of the most widely used programming languages (such as C++, Java, Python, etc.) are multi-paradigm and they support object-oriented programming to a greater or lesser degree, typically in combination with imperative, procedural programming. Significant object-oriented languages include: (list order based on TIOBE index) Java, C++, C#, Python, R, PHP, Visual Basic.NET, JavaScript, Ruby, Perl, Object Pascal, Objective-C, Dart, Swift, Scala, Kotlin, Common Lisp, MATLAB, and Smalltalk. Object-oriented programming uses objects, but not all of the associated techniques and structures are supported directly in languages that claim to support OOP. The features listed below are common among languages considered to be strongly class- and object-oriented (or multi-paradigm with OOP support), with notable exceptions mentioned.

Objects sometimes correspond to things found in the real world. For example, a graphics program may have objects such as "circle", "square", "menu". An online shopping system might have objects such as "shopping cart", "customer", and "product". Sometimes objects represent more abstract entities, like an object that represents an open file, or an object that provides the service of translating measurements from U.S. customary to metric.

Each object is said to be an instance of a particular class (for example, an object with its name field set to "Mary" might be an instance of class Employee). Procedures in object-oriented programming are known as methods; variables are also known as fields, members, attributes, or properties.

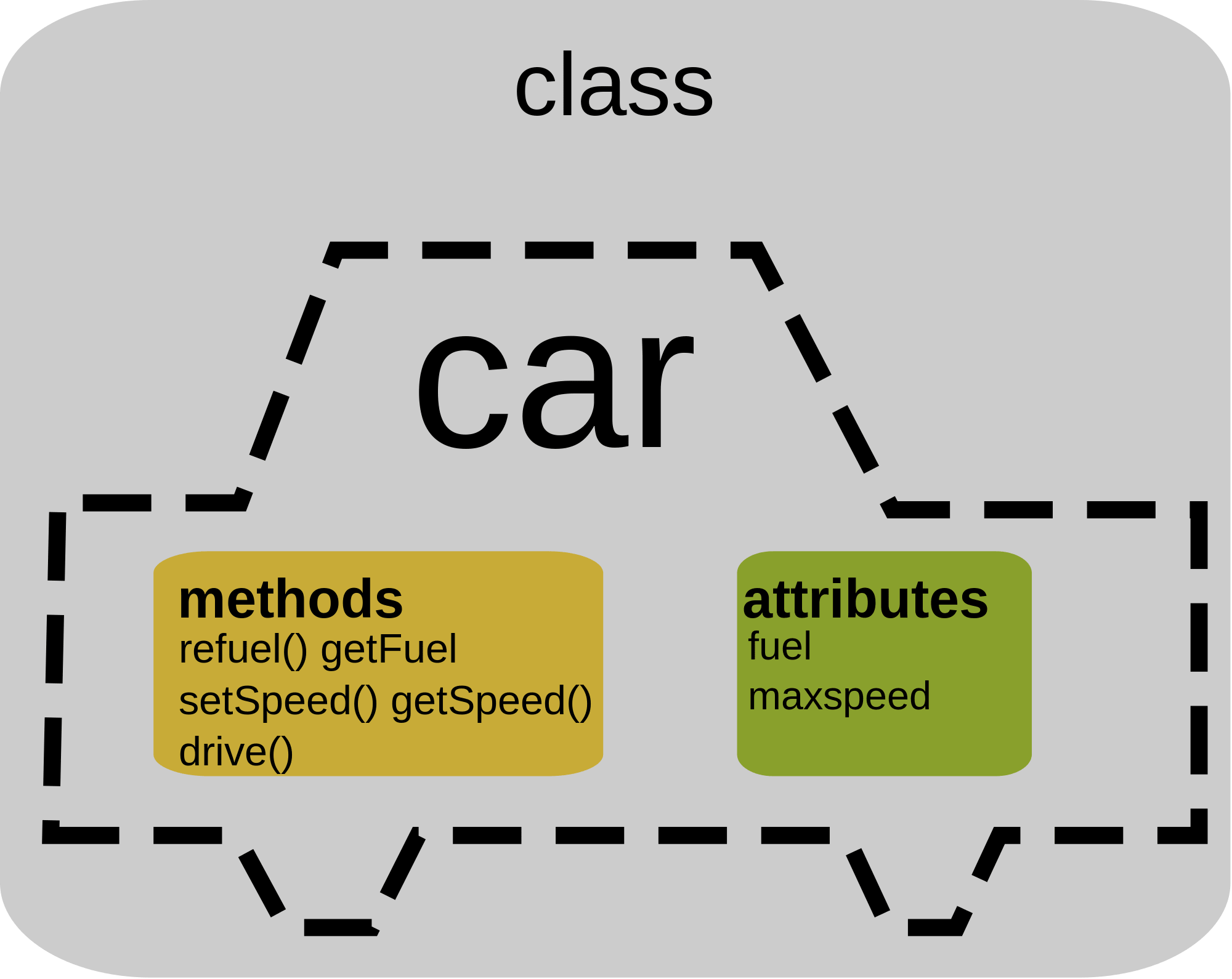


Fig. OOP Class and Object

**Conclusion and Future Enhancement**

1. **Conclusion:**

Objects are accessed somewhat like variables with complex internal structure, and in many languages are effectively pointers, serving as actual references to a single instance of said object in memory within a heap or stack. They provide a layer of abstraction which can be used to separate internal from external code. External code can use an object by calling a specific instance method with a certain set of input parameters, read an instance variable, or write to an instance variable. Objects are created by calling a special type of method in the class known as a constructor. A program may create many instances of the same class as it runs, which operate independently. This is an easy way for the same procedures to be used on different sets of data.

Object-oriented programming that uses classes is sometimes called class-based programming, while prototype-based programming does not typically use classes. As a result, significantly different yet analogous terminology is used to define the concepts of object and instance. In some languages classes and objects can be composed using other concepts like traits and mixins.

1. **Future Enhacements:**

In recent years, object-oriented programming has become especially popular in dynamic programming languages. Python, PowerShell, Ruby and Groovy are dynamic languages built on OOP principles, while Perl and PHP have been adding object-oriented features since Perl 5 and PHP 4, and ColdFusion since version 6.

The Document Object Model of HTML, XHTML, and XML documents on the Internet has bindings to the popular JavaScript/ECMAScript language. JavaScript is perhaps the best known prototype-based programming language, which employs cloning from prototypes rather than inheriting from a class (contrast to class-based programming). Another scripting language that takes this approach is Lua.

Object-oriented programming is a programming paradigm based on the concept of "objects", which can contain data and code: data in the form of fields, and code, in the form of procedures. A feature of objects is that an object's own procedures can access and often modify the data fields of itself.

**References**

Kindler, E.; Krivy, I. (2011). "Object-Oriented Simulation of systems with sophisticated control". International Journal of General Systems: 313–343.

Lewis, John; Loftus, William (2008). Java Software Solutions Foundations of Programming Design 6th ed. Pearson Education Inc. ISBN 978-0-321-53205-3., section 1.6 "Object-Oriented Programming"

Deborah J. Armstrong. The Quarks of Object-Oriented Development. A survey of nearly 40 years of computing literature which identified a number of fundamental concepts found in the large majority of definitions of OOP, in descending order of popularity: Inheritance, Object, Class, Encapsulation, Method, Message Passing, Polymorphism, and Abstraction.

John C. Mitchell, Concepts in programming languages, Cambridge University Press, 2003, ISBN 0-521-78098-5, p.278. Lists: Dynamic dispatch, abstraction, subtype polymorphism, and inheritance.

Michael Lee Scott, Programming language pragmatics, Edition 2, Morgan Kaufmann, 2006, ISBN 0-12-633951-1, p. 470. Lists encapsulation, inheritance, and dynamic dispatch.

Pierce, Benjamin (2002). Types and Programming Languages. MIT Press. ISBN 978-0-262-16209-8., section 18.1 "What is Object-Oriented Programming?" Lists: Dynamic dispatch, encapsulation or multi-methods (multiple dispatch), subtype polymorphism, inheritance or delegation, open recursion ("this"/"self")

Booch, Grady (1986). Software Engineering with Ada. Addison Wesley. p. 220. ISBN 978-0805306088. Perhaps the greatest strength of an object-oriented approach to development is that it offers a mechanism that captures a model of the real world.

Ali, Junade (28 September 2016). Mastering PHP Design Patterns | PACKT Books (1 ed.). Birmingham, England, UK: Packt Publishing Limited. p. 11. ISBN 978-1-78588-713-0. Retrieved 11 December 2017.

Jacobsen, Ivar; Magnus Christerson; Patrik Jonsson; Gunnar Overgaard (1992). Object Oriented Software Engineering. Addison-Wesley ACM Press. pp. 43–69. ISBN 978-0-201-54435-0.

McCarthy, J.; Brayton, R.; Edwards, D.; Fox, P.; Hodes, L.; Luckham, D.; Maling, K.; Park, D.; Russell, S. (March 1960). "LISP I Programmers Manual" (PDF). Boston, Massachusetts: Artificial Intelligence Group, M.I.T. Computation Center and Research Laboratory: 88f. Archived from the original (PDF) on 17 July 2010. In the local M.I.T. patois, association lists [of atomic symbols] are also referred to as "property lists", and atomic symbols are sometimes called "objects"