ISSN: 2321-7782 (Online)
e-ISJN: A4372-3114
Impact Factor: 6.047

Volume 5, Issue 3, March 2017
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Real time methodologies in Object oriented programming

M.V. Chowdary
Asst.professor
Computer science department
Chalapathi institute of engineering and technology
Lam, Guntur, India

Abstract: Object oriented programming strongly supports the concept of objects. A big effort has been spent on methodologies and techniques for specification, design and implementation of object-oriented systems but a few has been made on the classification of Object orientation aspects and its real time methodologies.

The main goal of this article is to classify the aspects like abstraction, encapsulation, inheritance and polymorphism and their inter-relation with real time methodologies.

Keywords: Reusability, Base class – Subclass, data hiding, abstraction – encapsulation, Inheritance, polymorphism.

I. INTRODUCTION

Object-oriented development methods are becoming more and more popular and object- orientation is viewed by many people as the silver bullet for solving software related problems like maintenance, reuse, and quality. In the last decade the research in this field has grown considerably. A lot of effort has been put in the definition of methodologies and techniques for the specification and design of object-oriented systems [2,3,4,5]. The term object-orientation has been applied to many subjects such as: analysis, design implementation, data modeling in databases, and distribution. Object-oriented programming is meant to cover all these subjects, since one of the advantages of object-orientation is that it provides a unified approach to these subjects.

II. OBJECT ORIENTATION ASPECTS

In literature several different definitions of object-orientation can be found and the term *object-oriented* is often used to describe a lot of different approaches and/or techniques. Object-oriented technology comprises specific analysis, design and implementation methods. In the following we introduce the characteristics we consider as essential for an object-oriented programming:

1. **Data Abstraction:** Objects are described as implementations of abstract data types (ADTs). An object-oriented language must support abstract data type definitions. Usually, an ADT definition is called *class*, while an *object* is a run-time instance of a class. Abstraction only shows relevant data and hides the irrelevant details from the user. Thus, Abstraction shows the essential features and hides the non-essential features to the user.

Example: Let us consider any mobile like Nokia, Samsung, Iphone

a) Dialing a number call some method internally which concatenates the numbers and displays it on the screen, but we are unaware of what is it doing internally.

b) Clicking on a mobile green button actually sends signals to calling person’s mobile but we are unaware of how it is doing.

Abstraction tells about what is visible to the user. This is called abstraction where creating a method which is taking some parameter and returning some result after some logic execution without understanding what is written within the method.

2. **Encapsulation:** encapsulation is defined as the process of enclosing one or more details from outside world through access right. It says how much access should be given to particular details. Both abstraction and encapsulation works hand in hand because, abstraction says what details to be made visible and encapsulation provides the level of access right to that visible details. Encapsulation implements the desired level of abstraction.

Consider an example, when mobile A is connected with mobile B via bluetooth, it doesn't mean that A can access all the features of B. Similarly when mobile A is connected with mobile B and mobile B is connected to C, that doesn't mean that mobile A can connect to mobile C via mobile B. This is because a restriction in accessibility.

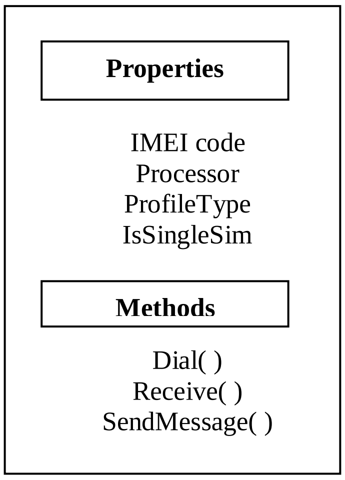
In oops Abstraction and encapsulation can be achieved through class and objects.

Objects: Any real world entity which can have some characteristics or which can perform some work is called as an object. Object is also called as an instance.

For example**,** any mobile manufacturing company at a time manufactures lak hs of pieces of mobiles of each model which are actually an instance. These objects are differentiated from each other via some identity or characteristics.

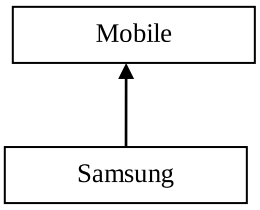
Class: A class is a plan which describes the object. It is a blueprint how object should be represented. A Class contains attributes and operations.

For example**,** a mobile can be a class which has some attributes like IMEI number, processor, profile type etc. and operation like Dial( ), Receive( ) and SendMessage( )

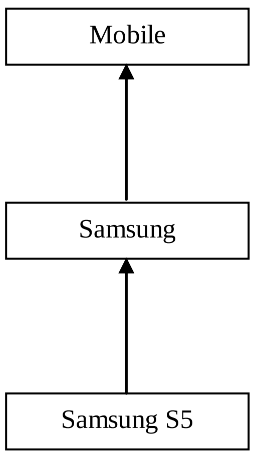


3. **Inheritance:** Inheritance is a key concept for object-orientation. Object-oriented languages must allow for defining an abstract data type deriving it from an existing one (either extending it or restricting it). The ability to extend the functionality from base entity in the new entity belonging to the same group. Thus we can reuse the used functionality. There are four types of inheritance

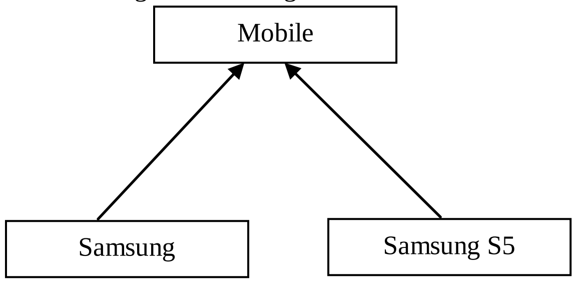
a) **Single inheritance:** Here there is a single base class and a single derived class**.** For example, base class mobile features are extended to Samsung brand.



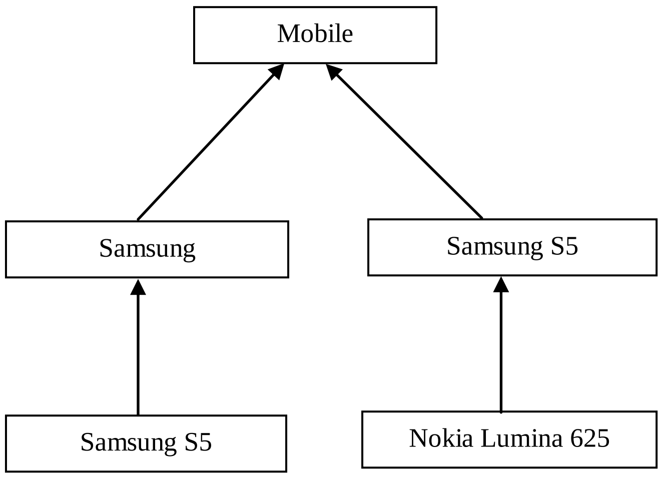
b) **Multilevel inheritance**: Here there is more than one single level of derivation. For example, base class mobile features are extended to Samsung brand and Samsung features are extended to Samsung S5.



c) **Hierarchical inheritance:** Here, multiple derived classes would be extended from base class. For example, base class mobile features are extended to Samsung and Samsung S5.



d) **Hybrid inheritance:** Single, multilevel and hierarchical inheritance altogether construct a hybrid inheritance. For example base class features are extended to



4. **Polymorphism:** The ability of doing the same operation but with different type of input. Program entities should be permitted to refer to objects of more than one class, when a hierarchical relationship among these classes exists. For example operator '+' can be used to add two numbers or it can also be used to concatenate two strings
5. **Dynamic binding:** Operations applied to a polymorphic variable should be permitted to have different realizations and the identity of such operations must be resolved dynamically based on the type of the object the variable is referring to.

References

1. Deborah J. Armstrong. The Quarks of Object-Oriented Development.
2. G. Booch. Object Oriented Design. Benjamin/Cummings Publ., USA, 1991.
3. G. Booch, I. Jacobson, and J. Rumbaugh. Unified Modeling Language User Guide. Addison-Wesley, 1997.
4. P. Coad and E. Yourdon. Object-Oriented Analysis. Prentice Hall, London, 2 edition, 1991.
5. J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. Object- Oriented Modeling and Design. Prentice Hall, Englewood Cliffs, 1991.
6. Pecinovsky, Rudolf (2013). OOP - Learn Object Oriented Thinking & Programming. Bruckner Publishing. ISBN 978-80-904661-8-0.
7. Rumbaugh, James; Michael Blaha; William Premerlani; Frederick Eddy; William Lorensen (1991). Object-Oriented Modeling and Design. Prentice Hall. ISBN 0-13-629841-9.
8. Schach, Stephen (2006). Object-Oriented and Classical Software Engineering, Seventh Edition. McGraw-Hill. ISBN 0-07-319126-4.
9. Schreiner, Axel-Tobias (1993). Object oriented programming with ANSI-C. Hanser. ISBN 3-446-17426-5. hdl:1850/8544.
10. Taylor, David A. (1992). Object-Oriented Information Systems - Planning and Implementation. John Wiley & Sons. ISBN 0-471-54364-0.
11. Weisfeld, Matt (2009). The Object-Oriented Thought Process, Third Edition. Addison-Wesley. ISBN 0-672-33016-4.
12. West, David (2004). Object Thinking (Developer Reference). Microsoft Press. ISBN 0735619654.