This executive summary will looks at the underlying principles of object-oriented programming. It will explain the benefits of these principles and also their limitations in brief.

Object-oriented programming(OOP) is a logical model that is used by programming languages so that the programs are centralized around structures of data that are called ‘objects’. The objects are, like in the case of Java, built in the form of ‘classes’.The objects themselves are instances of these classes. These objects contain data in the form of attributes, which are also called fields. Objects can be any form of real world object like animal or more intangible objects such as circles or squares. For example, an object of type House may contain several attributes or fields, one of which may be the number of bedrooms in that house. Objects interact with one another to facilitate various procedures that we may want to perform to meet our end goals.The four fundamental principles that help enable this are inheritance, polymorphism, encapsulation, and abstraction. These will be looked at in this summary, along with their benefits.

Inheritance allows the production of superclasses and subclasses. The objects belonging to a subclass ‘inherits’ the properties of the superclass. This means that any data relevant to the superclass in the form of fields and methods will also be applicable to the subclass in the form of it’s characteristics and behaviour. Subclass objects may even contain additional fields and methods that are not present in their superclass. This allows us to created specialized objects. For example, a superclass may have instances of type Property that defines the general properties of an available property and it’s subclasses may be defined based on the type of property, i.e., an apartment or a house. The subclasses not only inherit methods but may also modify or supplement them. Inheritance allows the reusability of the superclass code in the subclass. It also enables specialized objects that help structure data better and form a hierarchy of classes.

Polymorphism, as the name suggests, means many forms. Polymorphism is closely linked to inheritance. This functionality of OOP enables the existence of methods with the same name but different functionality. The same code can be used with different objects so that it can behave differently. The subclass implementation of a method means that the code will behave differently depending on the type of object that is used to call the method. This is distinguished by the runtime system when the code is run and is know as ‘late binding’. Therefore, polymorphism allows for flexibility and also supports the reusability of code along with inheritance. Polymorphism may also take place in the form of method overloading, where the same method are defined with different parameters, and, therefore, differ in their functionality.

Encapsulation is another fundamental concept. Through its implementation it is possible to create objects, restrict the visibility and access to their characteristics by defining the latter’s scope. This means that the object properties can be altered or accessed only by it’s own

methods. Encapsulation is also referred to in terms of data hiding. Data hiding can be implemented in two steps.Firstly, by specifying the scope of the fields of the class through ‘access specifiers’ - private, protected, public. Secondly, by defining methods that access or modify the fields. These methods are of two types, accessors and mutators. Getters are examples of accessors, while setters are examples of mutators and change the ‘state’ of an object. Thus, encapsulation enables us to protect the ‘integrity’ of the objects and stops users from changing the characteristics of a objects to make them malfunction. It stop data corruption through data hiding.

Abstraction work along with encapsulation to create objects. It is the design of the properties of an objects and a skeleton of how it is supposed to be implemented. Encapsulation essentially hides the implementation of the model that abstraction represents. Abstraction allows the implementation of a certain design at a later stage in the development process. This is aimed at reducing complexity.

Object-oriented programming also has it’s fair share of critics. It is criticised for its rigid structure and class hierarchies that limit flexibility through the strong relationships between superclasses and subclasses. Even methods are binding to the object(in the case of non-static methods), which limits their use to just those objects and goes against the principle of reusability of code. The emphasis on objects and structure compromises other aspects such as complexity.

References

Booch, G. (1994). *Object-oriented analysis and design with applications*. Redwood City, Calif.: Benjamin/Cummings Pub. Co.

Deitel, P. and Deitel, H. (n.d.). *Java how to program*.

Reges, S. and Stepp, M. (2011). *Building Java programs*. Boston: Addison-Wesley.