Report on the fundamental concepts of Object Orientation

**What is an object-oriented programming language?**

It is a programming model based on the concept of objects and classes. In this model, programmers define the functions that can be applicable to the data structures and their data type. Object-oriented programming turns data structure into an object, including both data and functions. It encourages the reusing of these objects in the same and other programs as well.

For example, we create a class ‘motorcycle' that represents all the properties a motorcycle has, such as color, model and brand name. In the next step, we create an instance of a motorcycle type object and can name it *my motorcycle* to represent a specific motorcycle. A motorcycle can only function after uniting multiple parts such as the handle, engine, wheels, headlight and other parts. So, here, each part combines with others to make a single object that contains different data attributes.

Fundamental Concepts of Object-Oriented Programming:

The four basic concepts of object-oriented programming are inheritance, polymorphism, abstraction and encapsulation. The following explanation of these four basic concepts can help you get better insights into object-oriented programming (OOP):

1. Inheritance

In object-oriented programming, inheritance is a mechanism where programmers can derive a class from another class. This concept of OOP can be useful in giving custom logic to existing frameworks and in declaring different exceptions. Inheritance also allows programmers to reuse previously written codes. This removes the burden of writing the same codes again, as programmers can make a derived class inherit the property of its parent class. For example, one can create two child classes and name them *hatchback* and *sedan* inherited from the parent class c*ar*. There are the following five different variations in inheritance in OOP languages:

* **Single inheritance:** It is the simplest form of inheritance where a class inherits only one parent class. Single inheritance enables code reusability and adds new features to the existing class.
* **Multiple inheritance:** When a class inherits more than one parent class, it becomes a multiple inheritance. As the child class inherits properties from different parent classes, it has access to all of its objects. It is different from a single inheritance property, as it allows an object or class to inherit from more than one object or class.
* **Multilevel inheritance:** When one class inherits properties from a derived class, it is multilevel inheritance. For example, class A extends class B and class B extends class C.
* **Hierarchical inheritance:** In this variation of inheritance, the different child classes inherit a single parent class. For example, a parent class C can have three subclasses, D, E and F.
* **Hybrid inheritance:** If there is a combination of more than one type of inheritance, it is a hybrid inheritance. It can be a combination of simple, multiple and hierarchical inheritances.

2. Polymorphism

Polymorphism is the core concept of the object-oriented programming language that allows programmers to build logical codes. In this concept of OOP, programmers can access objects of different types through the same interface where each type provides its own implementation of the interface. The following two types of polymorphism are useful in OOP:

* **Compile-time polymorphism:** Also called static binding polymorphism, it means binding occurs at compile time. Method overloading is an example of compile-time polymorphism. It allows programmers to use objects of the same name while their parameters can be different.
* **Runtime polymorphism:** Runtime polymorphism involves dynamic binding. Method overriding is an example of runtime polymorphism. In this process, an object binds with the functionality at the run time.

3. Abstraction

This is the core concept of object-oriented programming that allows programmers to be abstract or pick out common features of the objects and procedures. In other words, abstraction means hiding internal details and showing functionality. The primary aim of programmers behind using abstraction is to handle complexity by hiding irrelevant details. It is an extension of encapsulation.

For example, a customer may only use a few selections of tools like petrol, accelerator, clutch, brake, wheel and odometer of a motorcycle. Most of the engineering work remains inconsequential to the customer till it continues to function properly. To make a motorcycle run, a lot of pieces and parts work together. Letting the details and information get exposed to the customers can be a dangerous distraction. This is the concept of abstraction that is useful in many fields of engineering.

4. Encapsulation

It is the basic concept of object-oriented programming that helps programmers to bind the data and functions together that manipulate the data. Encapsulation of data leads to the OOP concept of data hiding and keeps them safe from outside attention. One of the common examples of encapsulation is a calculator, as anybody using a calculator understands its functions, but may not require an understanding of how it works inside. Encapsulation can help in hiding irrelevant details from the outside world and highlight the necessary characteristics of a class to the users.