The Characteristics of an Object Orientated Paradigm

Encapsulation is when all of the data and methods that work on the certain part of data are within one unit. This is often used to hide a lot of the long winded coding and to hide the internal representation or state of the object from the outside. The idea of the mechanism is not complex and can be simple. The reasoning behind this is to hide the attributes of the data. Which can make things easier when coming to find the code and also security as it cannot be seen from the outside users.

An example of Encapsulation is when you are coding in java or in any type of code that you create classes. You don't realise that you are creating encapsulation when making classes and indenting further code into that set class is storing the current state of that object with the set of methods using the attributes.

Polymorphism is the concept that objects of different types can be accessed through the same interface. They can all produce their own independent versions of this interface. So no matter which interface you are on you can get to everything.

There are many types of polymorphism:

Parametric polymorphism – Parametric polymorphism allows the function or a data type to be written on the bases that it can handle valuesuniformly without depending on their type. It is also a way to make a language more expressive while still maintaining full static safety

Dynamic polymorphism – this form of Polymorphism doesn't allow the compiler to determine the executed method. It will do it at runtime. A subclass can override a method of its superclass. That allows the developer of the classes to customise the behaviour of that set method

A Constructor is responsible for preparing the object for action, and in particular establishing initial values for all its data, i.e. its data members. Although it plays a special role, the constructor is just another member function, and in particular can be passed information via its argument list that can be used to initialise it. The name of the constructor function is the name of the class.
That's how C++ knows it's a constructor

Example:

```
MyClass *MyObjPtr = new MyClass ();
```

A Destructor is used to free the resources which are allocated in constructor by removing the main class that they are contained in but not deleting the data inside. It is also a user defined member function. It cannot return any value from its definition.
It gets called implicitly by compiler when lifetime of object is over according to the storage class that would have been set.
When an object is no longer needed it has to be deleted using

Example:

```
delete MyObjPtr;
```

An Abstract class is a template of methods and variables of a class that contains one or more abstracted methods. Abstract classes are used in all object-oriented programming languages such as Java, C++, C# and VB.NET

A Concrete class is a class that has an implementation for all of its methods that were inherited from abstract or implemented via interfaces. Also does not define any abstract methods of its own. This means that an instance of the class can be created/allocated with the new keyword without having to implement any methods first. Therefore, it can be inferred that any class that is not an abstract class or interface is a concrete class.

A base class is a class within an object-oriented programming language. Which other classes are derived from. It helps with the creation of other classes that can reuse the code inherited from the base class to build a steady foundation within the newly built class. A programmer can extend base class functionality by adding or overriding members relevant to the derived class

An object-oriented interface is all about the process of designing and creating an interface that the user can interact with and use they correct way and is built on object-oriented programming concepts.

Modern applications and operating systems that are built with an object-oriented programming language create user interfaces using object-oriented concepts.

Overloading is when two or more methods are in one class that have the same name but different arguments. This allows the user to reuse the same method name and not have problems. This is helpful when an input can be accepted by more than one argument triggering more than one outcome like multiple choice.

Overriding
Method overriding means defining a method in a child class that is already defined in the parent class with the same method signature, same name, arguments, and return type

Generic programming means that you are not writing source code that is compiled as-is but that you write templates of source codes that the compiler in the process of compilation transforms into source codes
A template is a piece of code that can be copied and modified to fit a specific situation.
Some templates show how, why, and when to use some feature of the language.
Some templates show how to implement design patterns.