**Abstraction**

Abstraction is the process of hiding implementation details by selecting data from a larger pool and information and only showing details relevant for functionality to the user. It helps to reduce programming complexity and effort. An example of abstraction is opening a Microsoft Word document. There is code written that prompts a computer to action when a user clicks the Word icon on their desktop/ start menu. However, the user does not need to know the implement details so only the essential information is shown.

**Encapsulation**

Encapsulation also known as data hiding, is the concept of wrapping variables and methods together as a single unit. The variables of a class can be declared private and hidden from other classes. These variables can only be accessed through the methods of their current class. A fully encapsulated class is created by making all the data members of the class private. Public setter and getter methods can be used to modify and review the data; control over the data is maintained as the class can be made read-only or write-only. Encapsulation enables the modification of the implemented code without breaking the code of other users, resulting in maintainability, flexibility and extensibility. An example of encapsulation is cars, all functions of a car are private to the owners with no one else having access to them.

**Inheritance**

Inheritance is the method by which a new class can adopt the properties of another. The inheriting class is a subclass/ child class and the original class is called the super/parent class. An example is a program for ecological classification, with an animal class. All animals identified in the written program will have common attributes/methods such as heterotrophy and mobility, however the animals can be classified further into species. These subclasses will have their own unique attributes whilst sharing properties with the parent class. The use of inheritance promotes code re-usability, it avoids the need to repeat code lines therefore contributing to code maintenance.

**Polymorphism**

Polymorphism is the ability of an object to take on many forms. There are two types of polymorphism.

**Method overriding**- declaring a method in a child class which is already present in the parent class. The method must have the same name and parameter as in the parent class. An advantage of method overriding is the child class can provide its own implementation of the method without modifying the parent code. A program can have a parent class called animal with child classes of dogs, cats and horses. The parent class may contain a method of noise, the child classes can override this method with specific behaviours of their own e.g. bark, meow and neigh. Method overriding is also called dynamic polymorphism because a call to the overridden method is resolved at runtime.

**Method overloading** – several methods present in a class have the same name but different parameters. The parameters must be different in either number, sequence or types of arguments. An example is the argument lists of method add (int a, int b) and add (int a, int b, int c); they have the same method but different parameters. At compile time, Java knows which method to invoke by checking the method signatures. Hence method overloading is also called compile time polymorphism. Method overloading increases the readability of a program.