**Exercise 1**

**Encapsulation**

Encapsulation is the act of hiding implementation details from an external user and providing them with an interface to interact with an object which allows an external user to use the object without knowledge of how it works. This also allows a strict control over how the user interacts with the object preventing the user from putting the object into a broken state.

More specifically, in Java, encapsulation keeps the variables of a class hidden, accessible only through the methods defined in that class. This is done by declaring the variables of a class as private and providing accessor and mutator methods to modify or view the variables values.

An example of this is if you have a class Animal with a method move, you don’t need to understand all the mechanics in how the object type Animal moves, you can just call the method with .move()

**Inheritance**

Inheritance is a way of linking similar objects under a parent object. A subclass, the child, will inherit behaviour from the superclass, the parent. This avoid writing duplicate code for each subclass by putting that piece of code in the superclass.

A class can only extend one other class, but a class can be extended by multiple class. This creates a hierarchical structure.

An example of an Inheritance structure is:

The superclass Animal is extended by the subclasses Mammal, Bird, and Fish. The Mammal class could then be further extended to Cat, Dog, and Horse.

**Polymorphism**

Polymorphism is the ability of a variable, function or object to take on multiple forms.

When a method is overridden in a subclass you create an inheritance structure where the same method may have different behaviours. this method is then said to be polymorphic.

An example of this is if you have a superclass Animals with a method move() which is inherited by subclasses Mammal, Fish, and Bird. With polymorphism, each subclass may have its own way of implementing this function. When move is called for an object of the Fish class, swim may be printed. When move is called for an object of the Bird class, fly may be printed.

**Abstraction**

Abstraction is a way of viewing a problem that focusing on the concept of the problem whilst ignoring any details of how this may be implemented. It defines an object in terms of its properties, functionality, and interface. You can have different levels of abstraction with high levels focusing on a larger scale ignoring more details and lower level focusing on smaller scales with more details.

In implementation, you have abstract classes that may contain abstract methods. You cannot define an object using an abstract class as it is used to ensure certain behaviours of subclasses.

The abstract method reduces the method to a concept only contains the name of the method, what it takes in as input, and what it outputs. The abstract method guarantees that the subclass has a section of code implementing the method. What that method is implemented is unimportant so long as it takes it the same inputs and returns the same output.

An example of this is if you have an abstract class Mammal with subclasses Horse, Bat and Dolphin and an abstract method move. Since each of these classes move in a different way the way in which the move method is implemented is different but we want all the methods to have the same purpose. The abstract method required each subclass to implement their own method move that follows the method signature given.