**Apprentice Project 1 – Task 1**

The four key principles that are associated with Object Oriented Programming are:

* **Abstraction**
* **Encapsulation**
* **Inheritance**
* **Polymorphism**

1. **Abstraction:**

Abstraction strips everything down to its **basic principles**. Abstraction is useful in OOP because we take out unnecessary details of an object and only focus on aspects that are necessary to that context, to improve efficiency. For Example in my program I have an object of “ParkingTicket” that is created and all of its attributes are in its own class. Such as arrival time, expiry time, whether the ticket is pre-paid and the registration number of the car.

1. **Encapsulation:**

Encapsulation is the principle of **hiding** information. As in, the internal workings of an object are hidden from the rest of the program. An object has to provide its users only with the essential information for manipulation, without the internal details. This is to reduce confusion and improve readability to both the programmer and anyone else who may want to view it.

To improve maintainability and readability of my program I used getters and setters. I have a class that stores all the ticket information and passes all information with getters which can be called from any other class as long as there is an instance of that class.

1. **Inheritance:**

Real-life objects have a certain amount in common with each other. Mountain bikes, road bikes, and tandem bikes, for example, all share the characteristics of bicycles (current speed and current gear). Yet each also have their own unique features. Tandem bicycles have two seats and two sets of handlebars; road bikes have drop handlebars; some mountain bikes have an additional chain ring, etc.

Inheritance in programming means to allow classes to **inherit commonly used behaviours** (methods or variables) from other classes. In the bike example, Bike is the “superclass” and MountainBike, RoadBIke and TandemBike are all “subclasses”. Inheritance helps reduce code redundancy by putting what the objects have in common and putting them into one place.

1. **Polymorphism:**

Polymorphism is the ability of an object to take on many forms. Polymorphism enables programmers to deal in generalities and let the system handle the specifics, which trims down the work of the developer.

As an example, the superclass Ticket could have subclasses Display and CalculateCost. The superclass has a function for Time and is inherited in the Display and CalculateCost subclasses. With polymorphism, each subclass may have its own way of implementing the function. So, for example when the Time function is called in an object of the CalculateCost class, the function might respond with the cost of parking for that length of that time. When the same function is called in an object of the Display class, the console might output “Duration of Stay: “ and then the display the time.