The five object orientated development ideas in which I am going to describe, compare and contrast are encapsulation, inheritance, polymorphism, aggregation and information hiding.

The object oriented development concept of Encapsulation is to hide your data. When a user is using a product they aren’t required to know about how the software works in the background. When someone is driving down the road in their car, they do not need to know how everything in their car from the engine to the radio functions. The purpose of Encapsulation is to hide the variables and methods of a class from the user so they cannot be accessed directly by said user outside of the class in which the methods/variables were created in, so the internal components of the class are invisible to the user as they are originally declared as private methods or variables. To access these hidden methods and variables, we are required to create accessor and mutator methods in which can get and set the variables. When using a getter method, the value of the variable stored in the class can be retrieved by calling the getter method. To change the value of the variable we need to call a set method and provide a value for the variable. To determine the level of access the user can have for the methods or variables, we can use different access modifiers such as private, protected, public and no modifier. These access modifiers determine the level of access the user can have.

Another object oriented programming concept is inheritance, the purpose of inheritance is to build relationships between classes, Inheritance allows the programmer to make a parent class(super class) in which another class can be derived from the parent class which can be called a child class(subclass). Inheritance allows a subclass to inherent the properties of existing objects in which are visible to the child class from the superclass. To determine and understand the super class to sub class relationship ‘is a’ can be used, for example a banana ‘is a’ fruit, therefore a banana class would inherit from the fruit class, if a ‘is a’ relationship doesn’t exist between the two classes, inheritance shouldn’t be used. In comparison with Encapsulation, inheritance allows a subclass to inherit methods and variables in which are accessible by the subclass. Inheritance allows the programmer to reuse code which makes the programme clearer and requires less space to function, whereas encapsulation is used for hiding certain aspects of a class by wrapping the data from the user as they aren’t required to know what the methods and variables are. The user may only be required to know which methods or variables which they are required to call for the process they want to run to function.

Polymorphism is a term in which an object can occur or have several forms which all differ from each other. Polymorphism allows methods to be created multiple times in which all slightly differ from each other which allows the compiler to distinguish between the methods, the compiler deciphers which method is which through the parameters provided to the method and the data types of each parameter. Method overloading and method overriding are both types of polymorphism. Method overloading is one method which is passed varieties of different arguments causing the method to behave differently on each instance. Method overloading is a static mode of polymorphism meaning, the compiler decides on compilation which method will be invoked. Polymorphism can also be dynamic which is referred to as method overriding. Method overriding means that a child class(subclass) is implementing a method which already exists within it’s parent class, where this differs from method overloading is that method overriding has the same method name, same parameters and the same return data type. Method overriding doesn’t directly affect the method within the parent class, it only affects the instance of the method created in the subclass provided the method in the parent class isn’t declared as protected or public. Method overriding requires different classes whereas method overloading normally requires only one class. Polymorphism differs from inheritance as inheritance allows the programmer to reuse their code, whereas polymorphism is the creation of multiple forms of one method. Inheritance directly affects the class by extending one class to another and exchanging methods through access modifiers and super methods, whereas polymorphism only affects certain methods and functions in which are called multiple times sometimes the purpose is to overload the method and other times the purpose is to override the methods. Polymorphism is directly linked to inheritance as dynamic polymorphism requires a parent class and child class to function. Encapsulation differs from polymorphism as encapsulation involves hiding the data by wrapping it and using defensive copies of objects whereas polymorphism involves a parent class providing methods and variables to a child class, while some of the methods and variables may not be allowed to be directly affected by the child class as the methods/variables may be classified as private, which only allows access within the class that they are created in.

The fourth concept of object oriented development which I am going to discuss is aggregation. Aggregation is a one way relationship between two classes, ‘has a’ is the relationship, for example a library ‘has a’ book, or a student ‘has a’ address. These relationships only work in one way as an address doesn’t have a student or a book doesn’t have a library. Aggregation is best used for reusing the code that the programmer wrote. Using references the programmer can link code through references to eliminate the use of re-writing the code multiple times throughout the project. Griffith college has an address, the college’s students and lecturers and staff members all have addresses as well, the code doesn’t need to be written multiple times, the programmer can make a reference of the address class for each the student, Griffith college and lecturers. Aggregation differs from inheritance as inheritance uses a ‘is a’ relationship and the classes are required to be a parent class or a child class whereas aggregation uses a ‘has a’ relationship. Inheritance allows all the methods in the parent class to be inherited by the child class whereas aggregation doesn’t require the classes to be parents or children of one another and doesn’t require the whole class to be accessed, while inheritance doesn’t require all methods to be directly accessible by it’s child the methods are still passed to the child.

Information hiding is the concept of hiding a classes methods and variables through the use of access modifiers which restrict how the data can be accessed by the user. This is useful when an external class may try to interact with certain methods of a class and interfere with it’s functionality, the internal features of the class could be hidden through a restrictive access modifier such as a private modifier, where methods created in a class can only be accessed within said class, protected is another restrictive access modifier in which the method can only be accessed within said class and also within it’s subclass and package, so only classes which extend the parent class or classes which are created within the package which the methods or variables are implemented in.