• Encapsulation

Encapsulation is wrapping variables into a class and making them private. It allows variables to be changed through functions which can allow data to be shielded as data is hidden and can only be accessed through the class.

An example of encapsulation would be:

A public class of Student which has variables such as name, age, student number, subject etc. This would be an example of encapsulation as it would have functions such as get and set for each variable that it has.

Example Source Code:

public class Student {

private String name;

private int age;

private int studentNum;

private String subject;

public void setName(String name){

this.name = name;

}

public String getName(){

return name;

}

public void setAge(int age){

this.age = age;

}

public int getAge(){

return age;

}

public void setNum(int studentNum){

this.studentNum = studentNum;

}

public int getNum(){

return studentNum;

}

public void setSubject(String subject){

this.subject = subject;

}

public String getSubject(){

return subject;

}

}

• Inheritance

Inheritance is the use of extends when creating classes ideally creating a subclass of an existing class. This allows the class that is extending to use functions and variables that the parent class uses and allows new variables or functions to be added on to the subclass.

An example of this would be:

A class of chair which has subclasses such as dining and office, these classes extend the chair class.

Example Source Code:

public class Chair {

public String material;

public int legs;

public Chair(String madeOf, int numOflegs) {

material = madeOf;

legs = numOflegs;

}

public void setMaterial(String newMaterial) {

material = newMaterial;

}

public void numOfLegs(int newNumofLegs) {

legs = newNumofLegs;

}

}

public class OfficeChair extends Chair {

public int seatHeight;

public OfficeChair(int startHeight,

String material,

int numofLegs) {

super(material, numofLegs);

seatHeight = startHeight;

}

public void setHeight(int newValue) {

seatHeight = newValue;

}

}

• Polymorphism

Polymorphism is the idea of having an object behave in multiple ways depending on the situation. There exist two types of polymorphism, one which is compile time and another which is run time. Compile time polymorphisms work via comparing the arguments that are given when calling a function and produces a result accordingly whereas run time polymorphism works where overrides and subclasses are used.

• Abstraction

Abstraction is the idea of concealing the inside details of a program. This allows for a less complex view of the program and helps us avoid code duplication. An example of this is encapsulation as variables are put into classes and are only accessed via methods that have been defined. The code above can be used as a model and an example of a less complex view would be setting the height of the Office chair as only the function will have to be called and the internal components of the function is not displayed.