# Excercise 1

## Encapsulation

Is the practice of keeping class fields private, allowing access only through public methods. It’s a protective barrier that avoids data being available program wide. It allows code to be modular, so it can either be rewritten or reused elsewhere.

Example

**class** car {

**private** String color = "white";

**public** String getColor() {

**return** color;

}

## }

## Inheritance

Inheritance allows sub/child classes to inherit properties such as fields and methods from super/parent classes. The keyword extends is used to define a child class.

Example:

A *car* and a *van* are both *vehicles*, they will have some of the same properties and should inherit from *vehicles*.

They will both have a *color* and a *number plate*; these are states/fields/variables.

They can both *accelerate* and *brake,* these are behaviours/methods/functions.

## Polymorphism

Allows properties in a class to be flexible. A polymorphic class has extended more than one class. An extended class may have a method of the same name as its parent, if called it will overwrite the parent method. A method may also have different functionality, depending on the arguments passed to it.

Example:

**class** Vehicle{

**public** **void** brake(){

System.***out***.println("The vehicle brakes");

}

}

**class** car **extends** Vehicle{

**public** **void** brake(){

System.***out***.println("The car brakes");

}

}

**class** Test{

**public** **static** **void** main(String[] args){

Vehicle vh = **new** car();

vh.brake(); // prints The Car brakes

vh = **new** Vehicle();

vh.brake(); // prints The Vehicle brakes

}

}

## Abstraction

Abstraction is about showing only relevant information. This is done by creating the appropriate structure of objects, classes, variables and methods. The structure will simulate the way we think about real objects, the purpose is:

* To aid readability.
* To avoid duplicate code.
* To hide the inner workings of methods.

Example:

The class *car*, does not need to know the inner workings of is *brake* method, only how to use it.

The class van also wants to use the *brake* method, so it would make sense to have them both use the *brake* method from a parent class *vehicle.*