# interfaces/abstraction

Abstract methods are implemented not overridden. it isn’t concretely defined. it will give you the goal, but the implemented methods will contain the logic. - it is basically a method without a body.

**Overloading/Overriding** → Overloading allows you to make the same method but take in different arguments and is independent to inheritance. Overriding is where the method in the child class changes the functionality of the method in the parent class to better suit its functionality and depends on inheritance. Overriding is used in Polymorphism.

Interfaces allow you to have loose coupling. which is good as tight coupling can stop you from making changes as it may other elements.

It allows you to separate what a class does from how it does it.

interface packages are always lower case

the convention is -able or I(then the name)

flexibility comes in where you can now implement more than one interface which solves the 1 class inheritance issue

when a class implements an interface for example ISwim, then it becomes of that type as well meaning that if you have an ArrayList<ISwim>, any object of a class with ISwim implemented can be added to that ArrayList. (Even if the class has multiple interfaces implemented it can still be added if one of them is the same, for example ICycle, ISwim, IRun.)

**Naming Convention** → Interfaces have a naming convention of -able or I -word such as runnable or IRun. if using I -word then I and the first letter of the word will be capital letters.

**Inheritance vs interfaces**

you use inheritance when you can see that the child classes are closely related and tightly coupled whereas interfaces are used to provide common functionality to unrelated classes.

allows you to be precise on the behaviours you classes will have as with inheritance all child classes will inherit everything from the parent class.

can have multiple interfaces vs a class can only inherit one class.