Interfaces I

What is an interface in Java?

What is an interface

An interface is a reference type pretty much like a class, but it can only contain constants, abstract method declarations, static methods (and actually default methods plus “nested types”, but that’s not for this course).

We can view an interface like a type which only declares a programming interface to a group of objects.

Interfaces are different from classes, in a number of ways.

They declare for instance what behavior (methods) are common to otherwise different types of objects.

An interface could be used to group unrelated stuff

One typical use for interfaces, is to define a common API (a common list of public methods) for some groups of classes which may be very different in all other aspects.

Often, interfaces have names like adjectives, such as “Comparable”.

It is perhaps easy to imagine that two completely different groups of classes share one common treat - they are all able to compare themselves to objects of the same type.

Book objects could be made comparable, as could Member or Passport objects. They have little in common other than being Comparable (to their likes).

Interfaces could also be viewed as contracts

Organizations could agree on a common behavior of some group of objects. The contract would be an agreement of a set of methods which should be present in every current and future implementation of this group of objects.

This contract could be expressed using a common interface, which lists the methods (in a general and abstract way) which every object should need to implement.

“All classes describing objects used to play some media file type should have a decode() method, as defined in the Decodable interface”

What does an interface look like?

An interface declaration in a “.java” file looks rather like a Java class, but it uses the `interface` keyword where a class declaration would use its keyword `class`.

```
public interface Decodable{

 /* declarations of abstract instance methods etc */

}
```

All instance methods are implicitly abstract in a method.

Interfaces are similar to abstract classes

Interfaces are quite similar to abstract classes with a few important exceptions.

An interface cannot declare constructors.

You can only extend one abstract (or concrete) class, but your class can declare that it implements many interfaces.

You use the `implements` keyword to say that a class implements (satisfies the contract of) an interface.

Implementing an interface

To say that your class satisfies the contract declared in an interface, you use the implements keyword.

If you write a concrete (non-abstract) class which implements an interface, your class must declare implementations of every abstract method declared in the interface.

If you don’t implement all methods in an interface, you must declare your class as abstract, since it then postpones the satisfaction of the contract to concrete subclasses.