What is an interface in Java?

Easiest way to think about it:
a very abstract class – all its methods are abstract.

Recall: The Abstract Animal

```
public abstract class Animal {
    public void sleep() {
        System.out.println("Sleeping: Zzzzz");
    }
    public abstract void roam();
    public abstract void makeNoise();
}
```

Typically, an abstract class provides implementations of some but not all of its methods...

Even more abstract Animal

```
public abstract class Animal {
    public abstract void sleep();
    public abstract void roam();
    public abstract void makeNoise();
}
```

... but it’s allowed to provide no implementation at all
Animal interface

```java
public interface Animal {
    public void sleep();
    public void roam();
    public void makeNoise();
}
```

Implementing an interface

Instead of `extends`, use keyword `implements`:

```java
public class Cow implements Animal {
    public void sleep() {
        some code
    }
    public void roam() {
        some code
    }
    public void makeNoise() {
        some code
    }
}
```

Implementing an interface is very similar to extending a class.

The implementing class must provide an implementation for every method of the interface.

Using an interface

You can use an interface as the type of a variable, just as you did with abstract classes. E.g. supposing cows moo,

```java
Animal myAnimal = new Cow();
myAnimal.makeNoise();
```

 does exactly the same thing, whether Animal is an abstract class or an interface.

As the client, you typically won’t care whether the general type is an abstract class or an interface – it makes no difference to you.

Either lets you, for example, write a method that returns an `Animal`, while encapsulating what kind of animal it is, if you wish.

Interfaces vs abstract classes

If interfaces behave so much like completely abstract classes, why does Java have a separate concept of interface?

Basically because multiple inheritance is such a pain!

In C++, one class can inherit from several superclasses. This turns out to be really problematic, theoretically and practically.

So in Java, each class has only one direct superclass.

**But** if the superclasses you want to inherit from are all completely abstract, then the problems don’t arise. So people wanted to be able to do that.

Solution: a class is allowed to implement several interfaces.
Where you'll mostly see this: collections

HashMap implements the interface Map
ArrayList implements the interface List
etc. – have a browse.