Interface

http://www.tutorialspoint.com/java/java\_interfaces.htm

Danil: Interface is a light version of abstract. Interface is just a template for writing a class, therefore it interface implemented by a class and NOT extended. Whereas abstract might contain code + template.

If a class does not perform all the behaviors of the interface, the class MUST declare itself as abstract.

* 1

An interface is a reference type in Java, it is similar to class, it is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface

* 2

Writing an interface is similar to writing a class. But a class describes the attributes and behaviors of an object.

* 3

Unless the class that implements the interface is abstract, all the methods of the interface need to be defined in the class.

* 4

An interface is similar to a class in the following ways:

1. An interface can contain any number of methods.
2. An interface is written in a file with a .java extension, with the name of the interface matching the name of the file.
3. The byte code of an interface appears in a .class file.
4. Interfaces appear in packages, and their corresponding bytecode file must be in a directory structure that matches the package name.

However, an interface is different from a class in several ways, including:

* You cannot instantiate an interface.
* An interface does not contain any constructors.

1. All of the methods in an interface are abstract.
2. An interface cannot contain instance fields.
3. The only fields that can appear in an interface must be declared both static and final.
4. An interface is not extended by a class; it is implemented by a class.
5. An interface can extend multiple interfaces.

Interfaces have the following properties:

* An interface is implicitly abstract. You do not need to use the **abstract** keyword while declaring an interface.
* Each method in an interface is also implicitly abstract, so the abstract keyword is not needed.
* Methods in an interface are implicitly public.

When implementation interfaces there are several rules:

* A class can implement more than one interface at a time.
* A class can extend only one class, but implement many interfaces.
* An interface can extend another interface, similarly to the way that a class can extend another class.

An interface can extend another interface, similarly to the way that a class can extend another class. The **extends** keyword is used to extend an interface, and the child interface inherits the methods of the parent interface.