Interfaces

New Austrian Coding School

Programming - memo

The more **abstract** we write our code, the more often we can **reuse** it. We can focus on **what** we want to achieve without caring about the **how**.

How to create an interface

○ We write **public interface** followed by the name ○ **Every** method will be **public**, so no need to add the keyword ○ We write the method signatures as usual ○ Instead of writing the method's body, we place a **semicolon** ○ Remember, we care about the ***what***, not the ***how***	**public interface** Hero { String getName(); **void** saveTheCity(); }

How to use an interface

○ We create a class as usual ○ We add **implements** plus the name of the interface ○ We write **at least** the code for the methods that the interface had ○ The @Override annotation indicates that we are replacing the method from the interface	**public class** Batman **implements** Hero { @Override **public** String getName() { **return "Batman"**; } @Override **public void** saveTheCity() { *// use his brain together with fancy technology* } }
○ We create an object from the **original** class, its **implementation** ○ We reference it with an **interface** ○ We **don't** need to know *who* is doing what, or *how* ○ We just need to know that what we want will be done	**public static void** main(String[] args) { Hero batman = **new** Batman(); Hero superman = **new** Superman(); List<Hero> heroes = Arrays.*asList*(batman, superman); **for** (Hero hero : heroes) { System.***out***.printf(hero.getName()); hero.saveTheCity(); } }