

Interfaces – The Hierarchy of Interfaces Using Extends



Implementing ipod ...

The screenshot shows the Eclipse IDE interface. A context menu is open over a Java class, listing several actions: 'Add unimplemented methods', 'Make type 'ipod_classic' abstract', 'Rename in file (Ctrl+2, R)', and 'Rename in workspace (Alt+Shift+R)'. A yellow tooltip is displayed on the right side of the menu, titled '8 methods to implement:', listing the following methods: package_1.mp3_player.play(), package_1.mp3_player.stop(), package_1.mp3_player.ff(), package_1.mp3_player.rew(), package_1.mp3_player.volume_up(), package_1.mp3_player.volume_down(), package_1.ipod.download(), and package_1.ipod.volume_up(). The background shows a list of markers for the class 'ipod_classic'.

Interfaces – The purpose of @Override



When Eclipse wrote the necessary methods into the previous class, it placed a symbol of @Override above each one.

The significance of this is the Developer/Eclipse marking the method as a copy / override of the Interface's methods.

The @Override is not strictly necessary but is a way of enforcing that the Developer is changing an Interface's methods, placing the @Override above a new method results in an error ...

```
4
5  @Override
6  The method play1() of type ipod must override or implement a supertype method
7      // TODO Auto-generated method stub
8
```