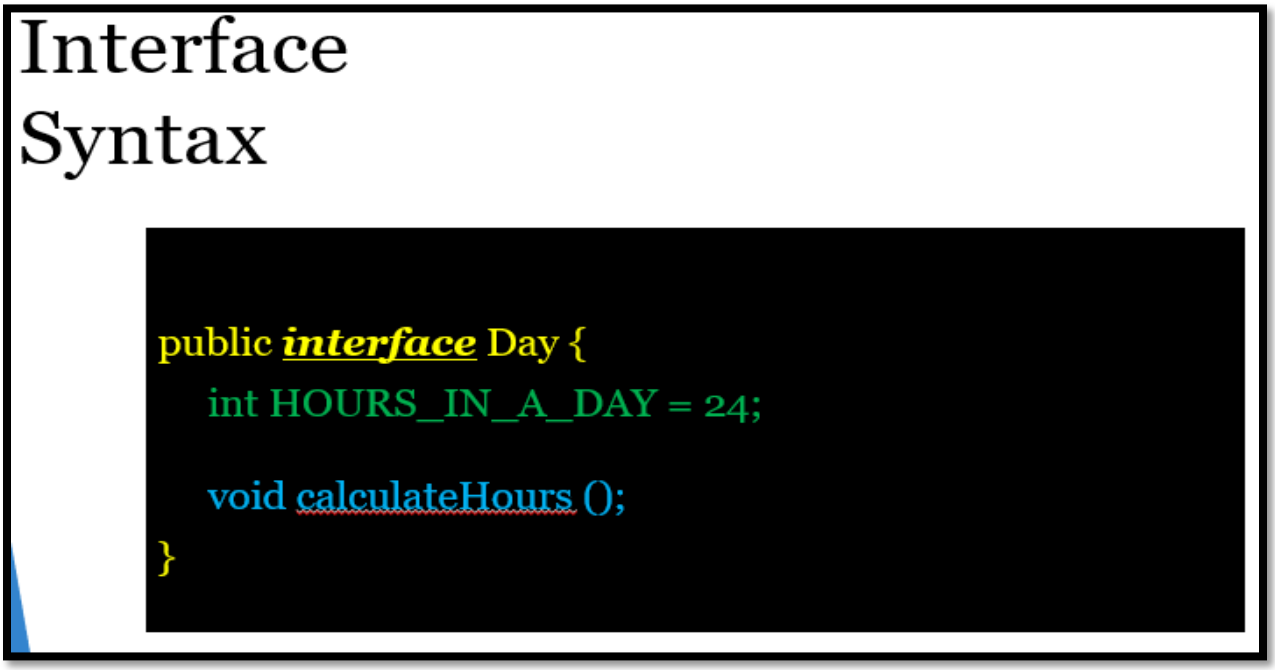
Rex Jones II

(Transcript) Java Interface Introduction

An interface is a contract with a class. Contracts are agreements for doing something or not doing something. So, the contract between an interface and class is the interface specifies what a class must do but not how the class must do it. Here's the syntax for an interface. All interfaces have the keyword interface before the interface name. The scope must be public or package. In this example, the scope is public but if we do not define a scope then the default scope is package.

Every field inside in an interface must have a constant value. Therefore, the variable is implicitly public, static, final, and requires a value. In this example, I did not include those 3 keywords: public, static, or final. However, Java will still treat int HOURS_IN_A_DAY = 24 as a constant because interfaces are contracts that cannot hold state. That means it is not allowed to change its behavior. So, we should not add a variable if the value is going to change.

When it comes to methods, they can be abstract, static, or default to an implementation. Notice this method 'void calculateHours' does not include the abstract keyword and it does not include a body. It does not include abstract or a body because the interface assumes all methods are abstract. As a result, the abstract keyword is not required.



Since the interface has a contract with the class, there is a guarantee that each abstract method listed in an interface will be implemented by the class. Static methods and default methods are not required to be implemented by a class because they already have an implementation.

Two of the benefits for using an interface, is to bring consistency for all classes that implement an interface and the support of multiple implementation. My plan is to demo both benefits by starting with consistency for all classes.