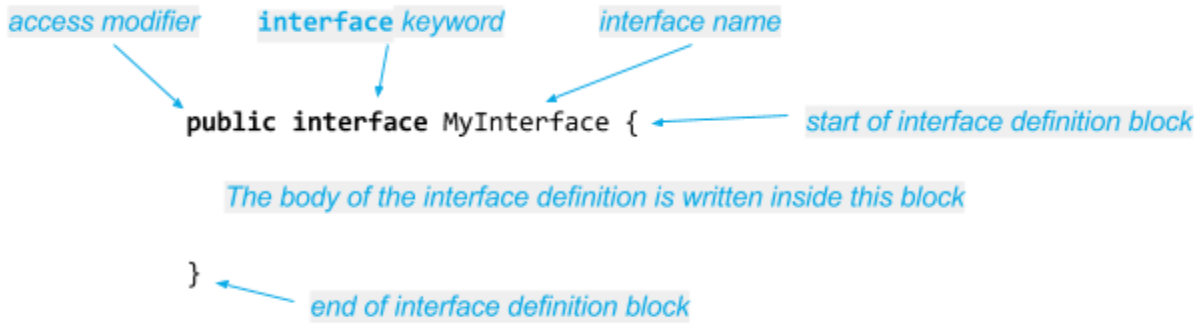
Quick Reference - Java Interfaces

Defining an Interface
Methods
Static Methods
Variables
Classes Implementing an Interface

Defining an Interface



access modifier - The `public` access modifier indicates that all code may use this interface. If an access modifier is omitted, the interface is assigned default access and is only accessible to other members of its package.

interface keyword - The `interface` keyword indicates that we are defining a new interface.

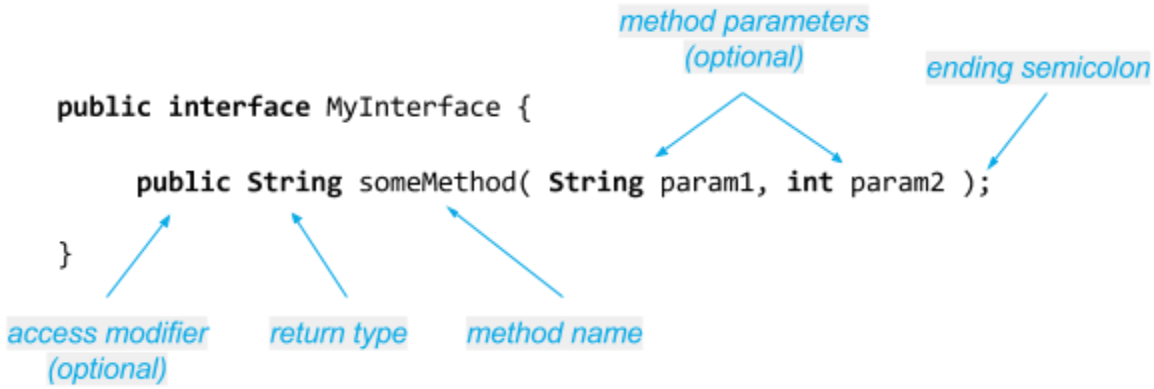
interface name - The name of the interface being defined. This name will be used when defining variables of this interface type and by classes implementing this interface. The rules for Java interface names are the same as those for any identifier:

- may be of any length
- may contain any unicode letter (e.g. a-z A-Z), number (e.g. 0-9), currency symbol (e.g. $) or underscore
- may not start with a number

By convention, interface names start with an uppercase letter and use camel case. Although currency symbols are valid characters for Java identifiers, by convention they are only used for system generated code and you should not use them in your interface names.

Copyright 2015 Tech Elevator LLC

Methods



access modifier - The `public` access modifier is optional. Interface methods are implicitly `public` so if the modifier is omitted, the method is still `public`.

return type - All method declarations must define a return type. If the method does not return a value, then the return type must be `void`.

method name - The name of the method being declared. This name will be used to invoked the method and by implementing classes to define a method implementation. The rules for method names are the same as those for any identifier:

- may be of any length
- may contain any unicode letter (e.g. a-z A-Z), number (e.g. 0-9), currency symbol (e.g. $) or underscore
- may not start with a number

By convention, method names start with a lowercase letter and use camel case. Although currency symbols are valid characters for Java identifiers, by convention they are only used for system generated code and you should not use them in your method names.

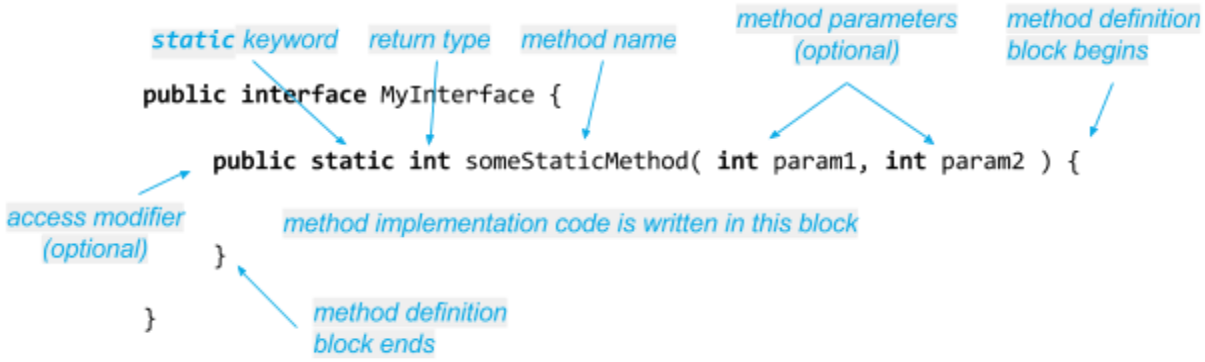
method parameters - Methods may optionally define one or more parameters. Parameters are defined as variables and must declare a type and name. Multiple parameters are separated by commas. If the method does not accept parameters, the method name is followed by empty parentheses.

ending semicolon - Unlike method declarations in classes, interface method declarations must be ended by a semicolon.

Copyright 2015 Tech Elevator LLC

Static Methods

(Java 8 and later)



access modifier - The `public` access modifier is optional. Interface methods are implicitly `public` so if the modifier is omitted, the method is still `public`.

static keyword - As is the case with static class methods, the static keyword indicates that the interface method can be invoked independently of any object. The method is instead called by referencing the interface name, followed by a period, followed by the method name. Example:

```
int x = MyInterface.someStaticMethod( 1, 2 );
```

return type - All method declarations must define a return type. If the method does not return a value, then the return type must be `void`.

method name - The name of the method being declared. This name will be used to invoke the method. The rules for method names are the same as those for any identifier:

- may be of any length
- may contain any unicode letter (e.g. a-z A-Z), number (e.g. 0-9), currency symbol (e.g. $) or underscore
- may not start with a number

By convention, method names start with a lowercase letter and use camel case. Although currency symbols are valid characters for Java identifiers, by convention they are only used for system generated code and you should not use them in your method names.

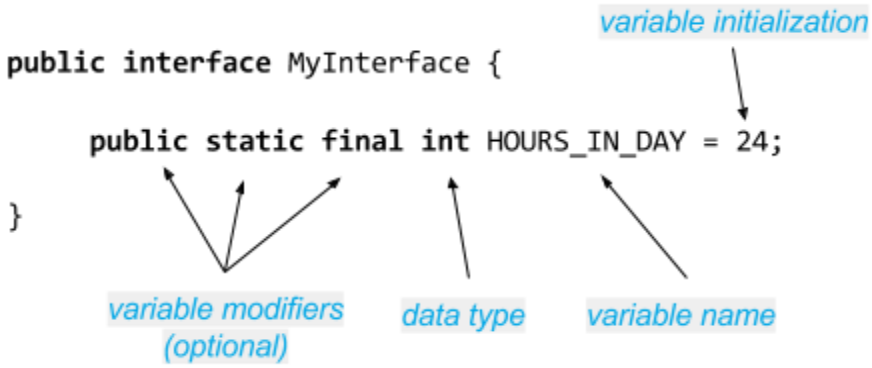
method parameters - Methods may optionally define one or more parameters. Parameters are defined as variables and must declare a type and name. Multiple parameters are separated by commas. If the method does not accept parameters, the method name is followed by empty parentheses.

Copyright 2015 Tech Elevator LLC

Copyright 2015 Tech Elevator LLC

Variables

(Java 8 and later)



variable modifiers - Interface variables are implicitly `public`, `static`, and `final` (i.e. they are constants). As such, these modifiers can be, and usually are, omitted from interface variable declarations. They are included here just for illustration purposes.

data type - All variables must declare what type of value they reference.

variable name - The name of the variable being declared. This name will be used by code that references this variable in an expression. The rules for variable names are the same as those for any identifier:

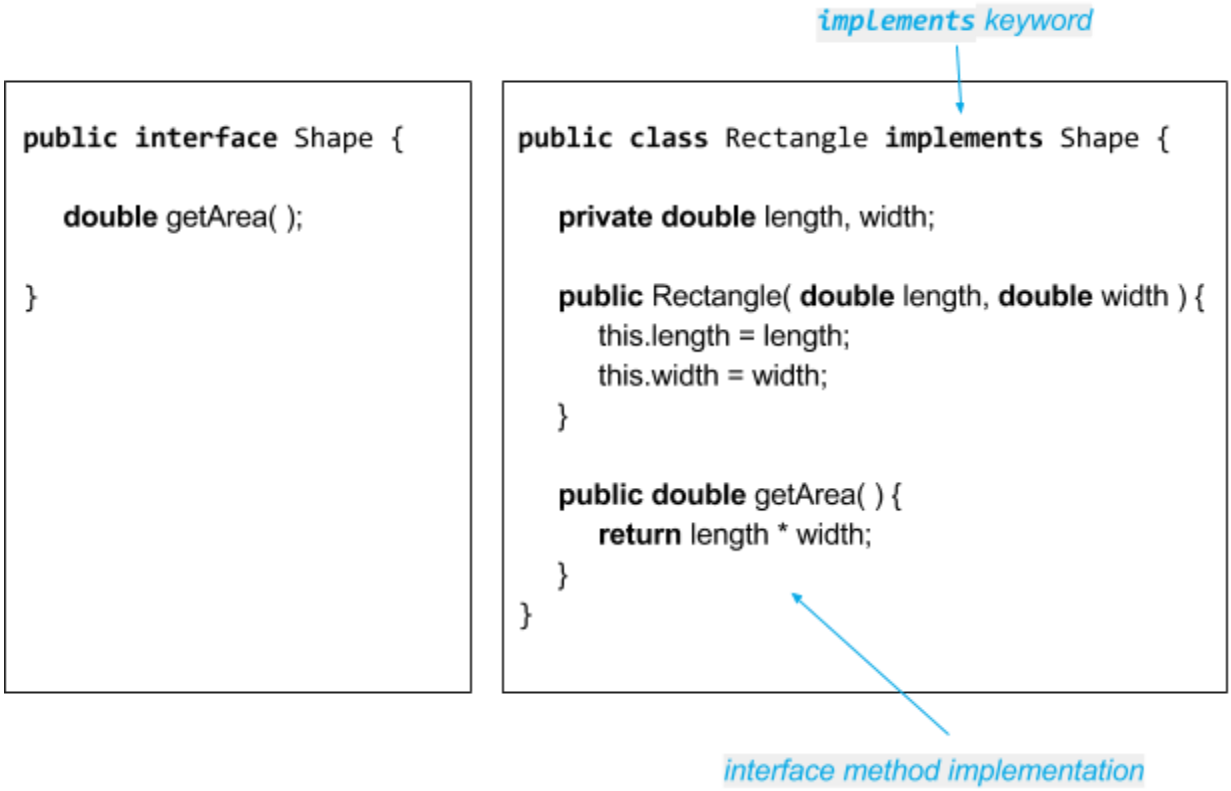
- may be of any length
- may contain any unicode letter (e.g. a-z A-Z), number (e.g. 0-9), currency symbol (e.g. $) or underscore
- may not start with a number

By convention, as constants, interface variable names use only uppercase letters and use underscores between words instead of camel case.

variable initialization - Because all interface variables are `final`, they must be initialized with a value as part of their declaration.

Copyright 2015 Tech Elevator LLC

Classes Implementing an Interface



implements keyword - The `implements` keyword appears to the right of the class name in the class definition to indicate that the class is an implementation of the interface whose name appears to the right of `implements`. A class may implement more than one interface by listing the name of each implemented interface to the right of the `implements` keyword with a comma between each interface name.

interface method implementation - A class must provide an implementation for each method declared by the interface(s) it implements. The method signature of the method implementation must match the method signature from the interface definition.

Copyright 2015 Tech Elevator LLC