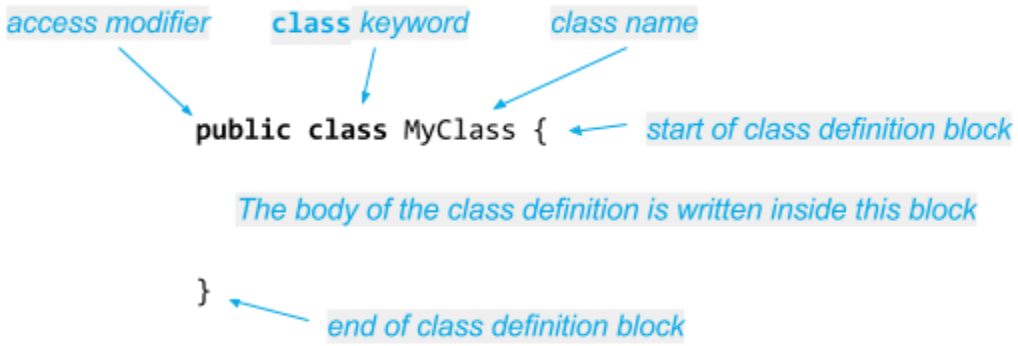
Reference Sheet - Java Classes and Objects

Defining a class



access modifier - Top-level classes (those that share the same name as the .java file they are defined in) are almost always defined as `public`. However, classes, especially inner classes, can be defined with access modifiers other than `public`. See Controlling Access to Members of a Class from the Oracle Java Tutorials for more information.

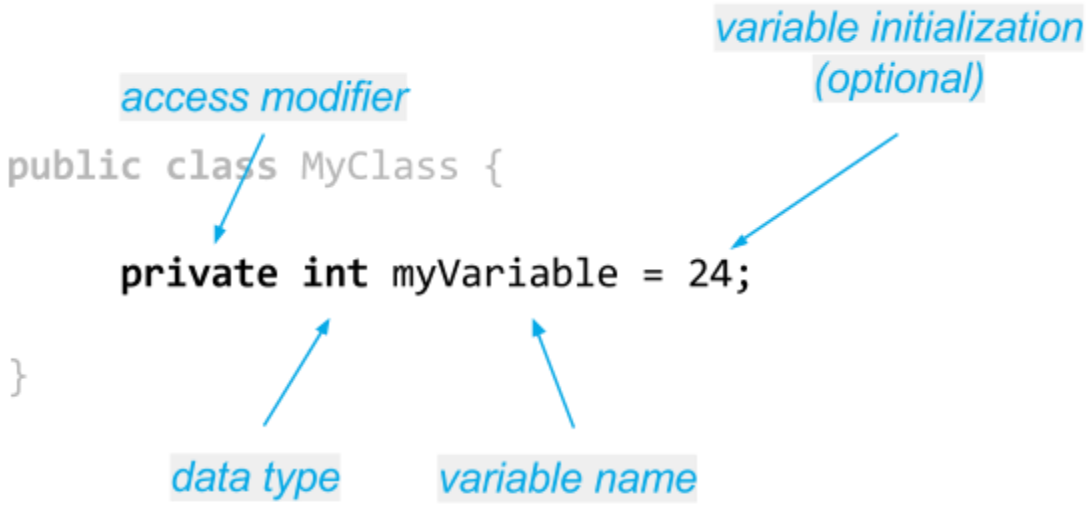
class keyword - The `class` keyword indicates that we are defining a new class.

class name - The name of the class being defined. This name will be used when defining variables of this class type. The rules for Java class names are the same as 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, class 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 class names.

Instance Variables



access modifier - Instance variables can be declared using either the `private`, `protected`, or `public` access modifier, or the access modifier can be omitted in which case the variable assumes default (i.e. "package private") access. See Controlling Access to Members of a Class from the Oracle Java Tutorials for more information.

You should typically declare instance variables as `private` unless you have a specific reason to provide broader access. Declaring instance variables `private` provides for better encapsulation of the internal workings of the class.

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

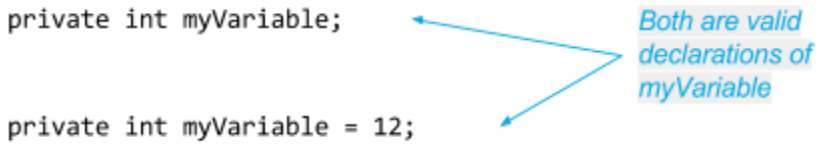
variable name - The name of the variable being declared. This name will be used by code that references this variable in an expression. The rules for variable names are the same as those for any identifier:

- 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, variable names will start with a lowercase letter and use CamelCase for any following words in the name. While the $ sign is a legal character it should be avoided in variable names.

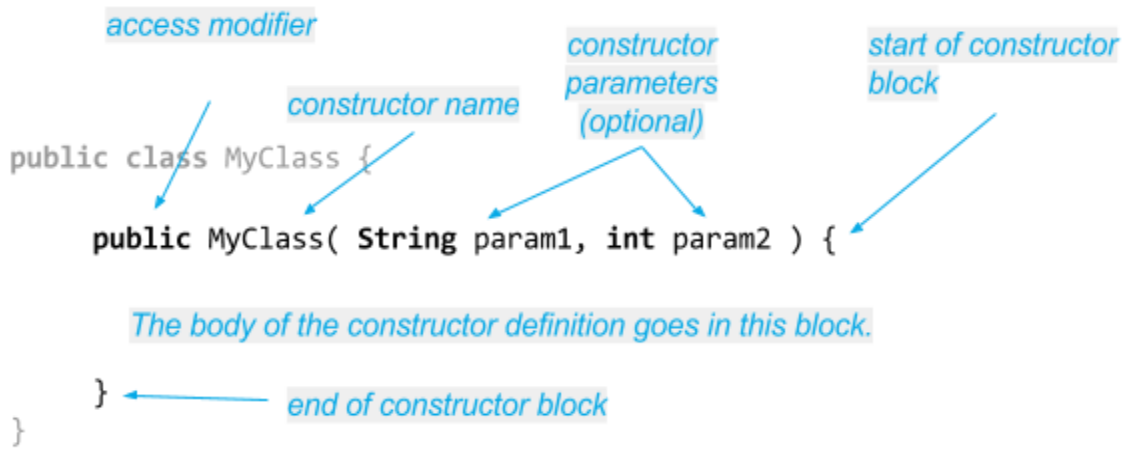
variable initialization - Like any variable, instance variables can optionally be initialized with a value when they are declared

Example:



If a variable is not initialized when it is declared, it can be initialized within a class constructor.

Constructors



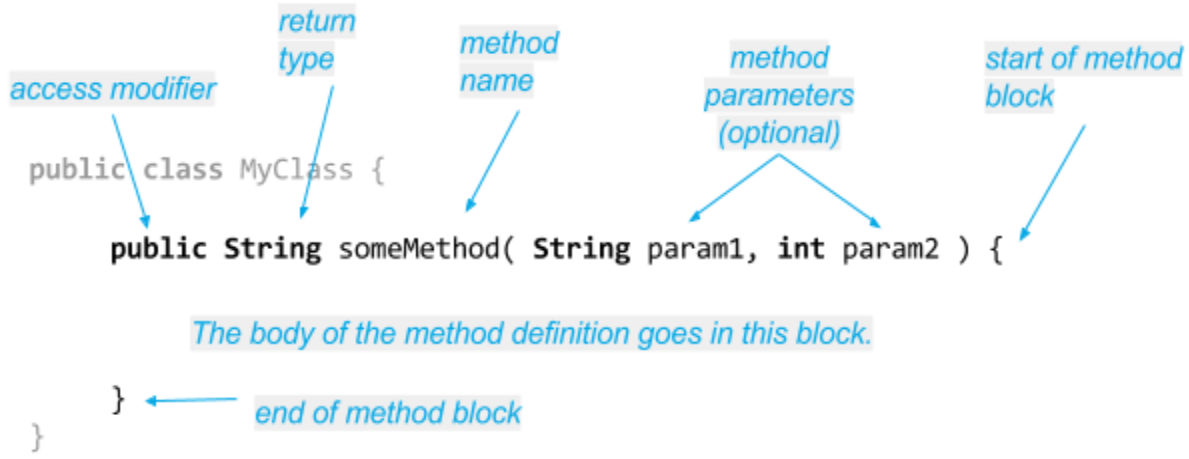
access modifier - Controls how the constructor can be accessed from outside the class. Can be set to `public`, `private`, or `protected`. Generally constructors are `public`. See Controlling Access to Members of a Class from the Oracle Java Tutorials for more information.

constructor name - The name of the constructor must be exactly the same as the class name. As such it should follow the naming conventions for class names.

constructor parameters - Constructors 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 constructor does not accept parameters, the constructor name is followed by empty parentheses.

Note: A constructor has no return type.

Instance Methods



access modifier - Instance methods can be declared using either the `private`, `protected`, or `public` access modifier, or the access modifier can be omitted in which case the method assumes default (i.e. "package private") access. See Controlling Access to Members of a Class from the Oracle Java Tutorials for more information.

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`. If a method declares a return type other than void, the method *must* return a value compatible with the declared return type.

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.