```
    (NSArray *)sprockets
```

This declaration says nothing about who should release the returned array. If the Gadget object returned an instance variable, it's responsible if the Gadget created an array and returned it, the recipient is responsible. This problem applies both to objects returned by a method and objects passed in as arguments to a method.

Ideally a body of code should never be concerned with releasing something it didn't create. The Foundation Kit therefore sets this policy: If you create an object you alone are responsible for releasing it. If you didn't create the object, you don't own it and shouldn't release it.

When you write a method that creates and returns an object, then, that method is responsible for releasing the object. It's clearly not fruitful to dispose of an object before the recipient of the object gets it, however. What's needed is a way to mark an object for later release, so that it will be properly disposed of after the recipient has had a chance to use it. The Foundation Kit provides just such a method.

(NSArray *)sprockets

```
(void)setMainSprocket:(Sprocket *)newSprocket
```

```
(void)setMainSprocket:(Sprocket *)newSprocket
```

```
(void)setMainSprocket:(Sprocket *)newSprocket
```

1].

containsObject:Find an object by repeatedly sending the array object an objectAtIndex: messag
incrementing the index until all objects in the array have been

The division of an interface between primitive and derived methods makes creating subclasses eas
override inherited primitives, but having done so can be sure that all derived methods that it inheri

The primitive-derived distinction applies to the interface of a fully initialized object. The question
should be handled in a subclass also needs to be addressed.

In general, a cluster's abstract superclass declares a number of init... and + className methods. As
Instancesº above, the abstract class decides which concrete subclass to instantiate based your choic
className method. You can consider that the abstract class declares these methods for the conveni
Since the abstract class has no instance variables, it has no need of initialization methods.

Your subclass should declare its own init... (if it needs to initialize its instance variables) and possi
methods. It should not rely on any of those that it inherits. To maintain its link in the initialization
its superclass's designated initializer within its own designated initializer method. (See the NEXT
Programming and the Objective C Language manual for a discussion of the designated initializers.
the designated initializer of the abstract superclass is always init.