**Annotations**

In Groovy, you can define and use annotations just like in Java.

Let’s have a look at one of those annotations that comes with the Groovy distribution: @Immutable.

import groovy.transform.Immutable;

@Immutable

class FixedCar{

String carName;

}

* A Groovy bean can be marked as immutable, which means that the class becomes final, all its fields become final, and you cannot change its state after construction.
* Above code declares an imuutablee FixedCar class, calls the constructor in two different ways:

def car = new FixedCar('Honda')

def car2 = new FixedCar(**carName:'Suzuki'**)

println(car.getCarName())

println(car2.getCarName())

* It has a standard implementation of equals() that supports comparison by content.
* With the help of try-catch, you assert that changing the state isn’t allowed.

def car = new FixedCar('Honda')

def car2 = new FixedCar(carName:'Suzuki')

assert car.carName == 'Honda'

assert car2.carName == 'Suzuki'

try {

car.carName = 'Maruti'

assert false, 'should not change the final property'

}catch(Exception e){

println("Error: $e.message")

}

* The @Immutable annotation not only make the class immutable, it also adds a correct hashCode()implementation and enforces *defensive copying* or access to all properties that aren’t immutable by themselves.
* Immutable types are always helpful for a clean design but they’re indispensable for *concurrent programming*.

***Using grapes***

* The @Grab annotation is used to explicitly define your external library dependencies within a script.
* We sometimes use the term *grapes* as friendly shorthand for our external Groovy library dependencies.
* In the Java world, you might store your dependent libraries in a lib directory and add that to your classpath and IDE settings, or you might capture that information in an Ivy, Maven, or Gradle build file. Groovy provides an additional alternative that’s very handy for making scripts self-contained. The following listing shows how you might use it.

@Grab('commons-lang:commons-lang:2.4')

import org.apache.commons.lang.ClassUtils

class Outer {

class Inner {}

}

assert !ClassUtils.isInnerClass(Outer)

assert ClassUtils.isInnerClass(Outer.Inner)