**1) Package and Imports:**

Packages are way to structure classes in different folders

* One problem with \* imports is that they can clutter your local namespace. But with the kinds of aliasing provided by Groovy, this can be solved easily.
* Groovy’s static import capability allows you to reference imported classes as if they were static methods in your own class:

import static Boolean.FALSE  
assert !FALSE //use directly, without Boolean prefix!

* This is similar to Java’s static import capability but is a more dynamic than Java in that it allows you to define methods with the same name as an imported method as long as you have different types
* Static import of method
* declaration of method with same name as method statically imported above, but with a different parameter type
* compile error in java, but is valid groovy code
* **If you have the same types, the imported class takes precedence.**
* Static imports with the **as** keyword provide an elegant solution to namespace problems. Suppose you want to get a Calendar instance, using its getInstance() method. It’s a static method, so we can use a static import. But instead of calling getInstance() every time, which can be misleading when separated from its class name, we can import it with an alias, to increase code readability:

import static Calendar.getInstance as now

assert now().class == Calendar.getInstance().class

**2) Groovy scripts :**

a script is any groovy code not enclosed in a class file but do not make a mistake thinking there is no class.

These scripts are compiled into class, the groovy compiler will compile the class for you, with the body of the scripts copied into a run method

**3) Classes:**

If it is groovy class then we can have same name for class as of file name, but if it is a script then we can’t have class name same as file name.

Anything out of class definition is considered as script

**4) Numbers :**

**In Java, if we create variables with primitive types, we cant invoke methods on it because they are not objects but primitive types but it is not the case with groovy**

1.getClass().getName();

Here 1 is not a primitive type but an instance of jav.long.integer

**5) Comment :**

//this is single line comment

/\* this is

A multiline comment

\*/

/\*\*

\* above 2 start is being used.

\*This is being used for documentation

\*/

6) **Assertion** : It is basically a statement; which enables you to test your assumption