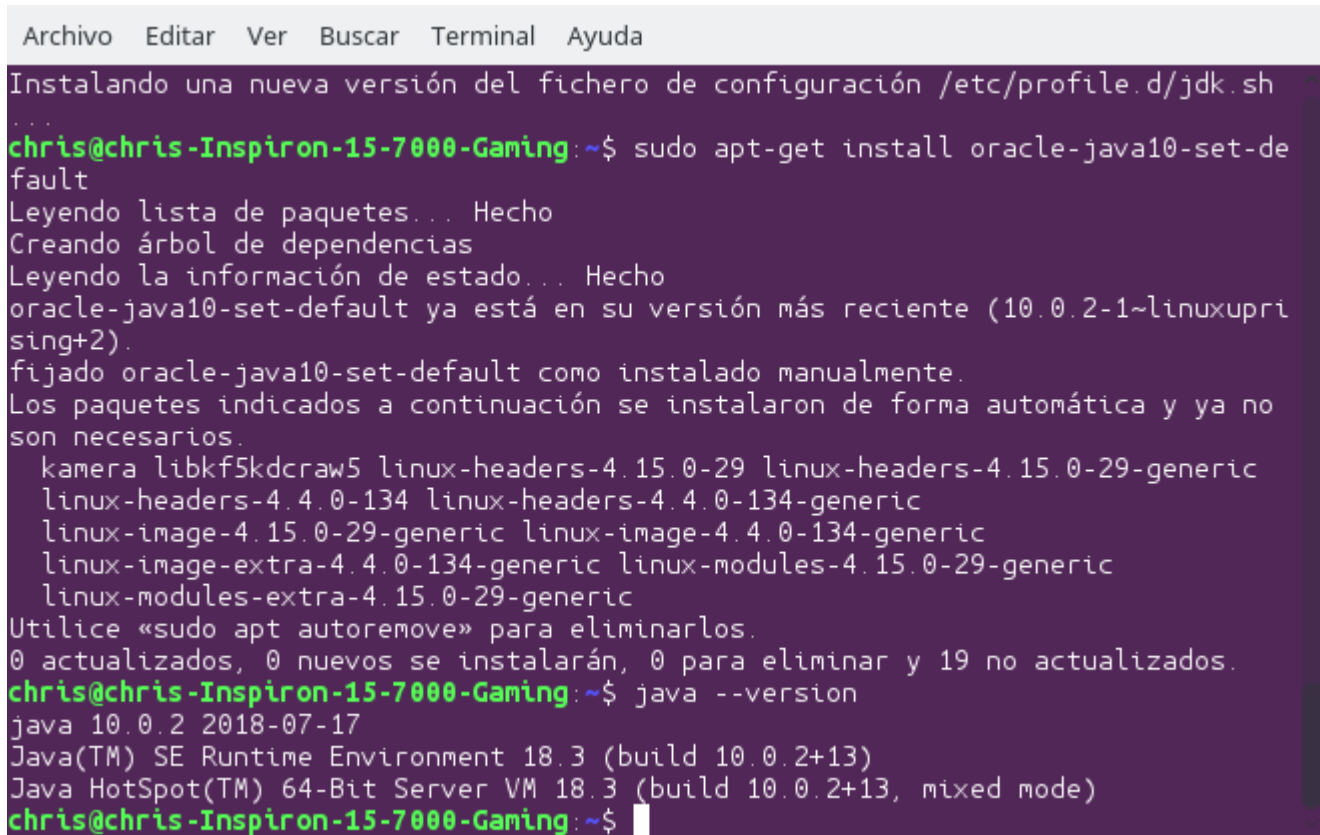


Para instalar necesitamos instalar una versión no más antigua que la 9 de JDK, para ello realizamos la siguiente verificación

java -version



```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
Instalando una nueva versión del fichero de configuración /etc/profile.d/jdk.sh
...
chris@chris-Inspiron-15-7000-Gaming:~$ sudo apt-get install oracle-java10-set-default
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
oracle-java10-set-default ya está en su versión más reciente (10.0.2-1~linuxuprising+2).
fijado oracle-java10-set-default como instalado manualmente.
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
kamera libkf5kdcraw5 linux-headers-4.15.0-29 linux-headers-4.15.0-29-generic
linux-headers-4.4.0-134 linux-headers-4.4.0-134-generic
linux-image-4.15.0-29-generic linux-image-4.4.0-134-generic
linux-image-extra-4.4.0-134-generic linux-modules-4.15.0-29-generic
linux-modules-extra-4.15.0-29-generic
Utilice «sudo apt autoremove» para eliminarlos.
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 19 no actualizados.
chris@chris-Inspiron-15-7000-Gaming:~$ java --version
java 10.0.2 2018-07-17
Java(TM) SE Runtime Environment 18.3 (build 10.0.2+13)
Java HotSpot(TM) 64-Bit Server VM 18.3 (build 10.0.2+13, mixed mode)
chris@chris-Inspiron-15-7000-Gaming:~$
```

En mi caso tenía una versión anterior es por ello que decidí en instalar esta nueva versión con los siguientes comandos

```
sudo add-apt-repository ppa:linuxuprising/java
sudo apt-get update
sudo apt-get install oracle-java10-installer
```

para después establecerlo como default

```
sudo apt-get install oracle-java10-set-default
```

Una vez hecho esto podemos arrancar la jshell con el comando 'jshell' y se termina el proceso con el comando '/exit'

Como sentencias, este programa acepta variables, métodos y definiciones de clase; importa y expresiones. Los fragmentos de código JAVA son reverenciados como snippets.

Para declarar una variable es tan simple como en java, sólo se declara el tipo que es y si se desea se inicializa

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

| Error:
| missing method body, or declare abstract
| String twice(String s);
| ^-----^

jshell> String twice(String s)
...> return s + s;
| Error:
| ';' expected
| String twice(String s)
| ^

jshell> String twice(String s){
...> return s + s;
...> }
| created method twice(String)

jshell> twice("Ocean")
$3 ==> "OceanOcean"

jshell> int x = 1
x ==> 1

jshell> 
```

Para declarar un metodo y clase, se hace como en un editor de texto, sólo que con la diferencia de teclear la apertura de llave justo en la primera linea, después de haber especificado los parámetros.

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

jshell> twice("Ocean")
$3 ==> "OceanOcean"

jshell> int x = 1
x ==> 1

jshell> class c {
...> int x;
...> }
| created class c

jshell> /list

 2 : String twice(String s){
   return s + s;
   }
 3 : twice("Ocean")
 4 : int x = 1;
 5 : class c {
   int x;
   }

jshell> 
```

Para modificar variables, metodos o clases se tiene que volver a definir la misma, por ejemplo

Archivo Editar Ver Buscar Terminal Ayuda

```
| Error:
| ';' expected
| String twice(String s)
|               ^
|
jshell> String twice(String s){
...> return s + s;
...> }
| created method twice(String)

jshell> twice("Ocean")
$3 ==> "OceanOcean"

jshell> int x = 1
x ==> 1

jshell> class c {
...> int x;
...> }
| created class c

jshell> /list

2 : String twice(String s){
```

Se define el método twice y en la siguiente captura se hace la modificación al mismo:

Archivo Editar Ver Buscar Terminal Ayuda

```
| illegal start of type
| String twice("thing")
|               ^
|
jshell> String twice("thing")
...>
...>
...>
...>
...> 1
| Error:
| illegal start of type
| String twice("thing")
|               ^
|
jshell> String twice(String s){
...> return "twice:" +s;
...> }
| modified method twice(String)

jshell> twice("thing")
$7 ==> "twice:thing"

jshell> |
```

Jshell acepta declarar metodos con referencias a otros métodos, variables o clases que aún no han sido definidas. Esto se hace para soportar la programación exploratoria y también porque algunas formas de programación lo llegan a requerir.

A continuación se cita el siguiente ejemplo:

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
...>
...>
...> 1
Error:
illegal start of type
String twice("thing")
^

jshell> String twice(String s){
...> return "twice:" +s;
...> }
| modified method twice(String)

jshell> twice("thing")
$7 ==> "twice:thing"

jshell> double vo
jshell> double volume(double radio){
...> return 4.0 / 3.0 * PI * cubo(radio);
...> }
| created method volume(double), however, it cannot be invoked until variable P
I, and method cubo(double) are declared

jshell> 
```

Se puede ver que se ingresó el método sin embargo nos pide que definamos los demás metodos dentro de este para poder regresar un resultado.

Una vez definiendo las variables y metodos necesarios nos regresa ya el resultado:

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
...>
...>
...> 1
Error:
illegal start of type
String twice("thing")
^

jshell> String twice(String s){
...> return "twice:" +s;
...> }
| modified method twice(String)

jshell> twice("thing")
$7 ==> "twice:thing"

jshell> double vo
jshell> double volume(double radio){
...> return 4.0 / 3.0 * PI * cubo(radio);
...> }
| created method volume(double), however, it cannot be invoked until variable P
I, and method cubo(double) are declared

jshell> 
```