**MI GUÍA PERSONAL DE LENGUAJE JAVA**

**By:... Beanswalker**

Esta guía la he construido a partir de mis experiencias con el Lenguaje de Programación Java y el IDE Netbeans Corriendo sobre GNU/Linux Deepin.

La Instalación y conficuración del Sistema Operativo, del IDE Netbeans con sus plugins y del OpenJDK está descrita en el Manual respectivo(Configuración de mi GNULinux para desarrollo.doc) que, por supuesto, está en Github.

**Lenguaje JAVA**

Empezamos nuestro aprendizaje del Lenguaje de Programación Java, el cual trabajaremos sobre el OpenJDK versión 11 y Utilizando mayoritariamente el IDE Netbeans y en casos muy aislados, el IDE Eclipse.

Como apoyos de aprendizaje. Vamos a utilizar:...

* Los cursos:...
  + Universidad de Java
  + Universidad de Java 2

Los utilizaremos Consecutivamente, ya que son del mismo autor y, en efecto, el 2 es la continuación. Están contenidos en mi Espacio de nube de MEGA.

* Los Libros:...
  + Cómo Programas en Java edición N° 10 de los hermanos Deitel (año 2016. Es lo que hay :P)
  + Programaci+on Orientada a Objetos Usando Java del Colombiano Hector Arturo Flórez Fernández (año 2012 porque también es lo que hay :P :P )

Java es un Lenguaje de programación que, en escencia es Programación Orientada a Objetos.

**Generalidades:..**

1. Todo el código que picamos de Java, se contiene en archivos que tienen la extensión .java
2. Al implementar Java, estamos trabajando con un lenguaje de programación que, aunque es basado en C y C++, trae muchas mejoras respecto a estos lenguajes y además es totalmente Orientado a Objetos. De manera que definimos clases para Modelar, organizar, manipular y presentar los datos de nuestra aplicación por medio del motor de las clases... Los métodos(Funciones y Procedimientos, que en POO pasa a ser llamados Métodos).
3. Accedemos al contenido de las clases mediante la implementación de Instancias - Objetos de las mismas
4. Java es un lenguaje Distribuído. Provee una colección de clases que permiten el funcionamiento de un programa distribuído en una red o ejecutar conexiones con servidores remotos, permitir las conexiones de clientes remotos
5. Java es multihilo per se, lo que permite ejecutar varias operaciones por separado, de manera simultanea y mantenerlas sincronizadas si son codependientes, guardando asi, la integridad de datos presentada como resultado.

El soporte al termino THREAD es ofrecido por Java mediante clases e interfaces definidas en el paquete java.lang y mediante métodos dedicados a la manipulación de Threads de una clase llamada Object.

Yo digo que es multihilo per se, porque no es necesario importar el paquete java.lang para utilizar sus clases en un proyecto, y porque Object es la clase Raiz, la clase madre o principal de la plataforma Java porque Todas las clases predefinidas y aún, las clases que nosotros implementamos, Heredan de la Clase Object.

1. La seguridad en Java está fundamentalmente en dos ámbitos:...

* Ámbito de desarrollo, el el cual el el JDK provee herramientas como:
  + Los programas “jar” y “javakey”, el primero es un empaquetador que premite la transferencia de datos de una manera compacta en archivos .jar. El segundo es un programa que permite el firmado de clases(¿generación de llaves?) de los archivos .jar
  + Una API para programación segura que introduce clases que, asu vez contienen métodos criptográficos para los programadores
  + Policy File:.. es un archivo o fichero donde se pueden definir políticas para asignar o restringir permisos de uso para acceso a recursos del sistema en todo el código
* Ámbito de ejecución, del cual se encarga específicamente la JVM. Ésta se encarga entre otras cosas de:...
  + Vigilar el cumplimiento de las normas de seguridad pre-establecidas
  + Verificar las referencias a Arrays
  + No permitir en algunos casos la conversión(Casting) entre distintos tipos de datos.
  + No hay manera de manipular directamente los punteros
  + Gestiona automáticamente el uso de la memoria, de manera que no queden “huecos”

Por ejemplo, en la ejecución de un código Java via Web, el navegador llama a la JVM, y le pasa los datos. La JVM carga las clases que se requieren para ejecutar el código, durante este proceso, se ejecuta un “verificador” en busca de código malicioso, antes de ejecutar el programa

1. Java utiliza un compilador tanto como Un interprete, ya que al ejecutarlo con el IDE por primera vez, el código es Compilado a un bytecode(código intermedio) por medio de un compilador llamado JavaCompiler, mas conocido como JavaC, convirtiendo nuestros archivos .java en archivos .class y posteriormente, cada vez que el mismo código de nuevo es ejecutado, el bytecode(todos los archivos .class) es interpretado en la JVM. Esta peculiaridad la vamos a comprender en la marcha.
2. Gracias a la JVM, Java es multiplataforma, y el mismo código escrito en X sistemaOperativo y compilado a bytecode, puede ejecutarse sin problemas y sin modificaciones en GNU/Linux u otro sistema Operativo que tenga instalada la JVM respectiva
3. El lenguaje de programación Java puede implementarse en muchos ámbitos de la tecnología informática (en componentes electrónicos, programas para electrodomesticos, en desarrollo de software a nivel empresarial para sistemas operativos o para la Web, así como para dispositivos móviles y en varios escenarios mas).

De manera que Java puede implementarse en forma de:...

* Componentes(conocidos como JavaBeans)
* Conexiones persistentes a BD(JDBC o Java DataBase Connectivity )
* Backend completo de proyectos Web
* Manejo de redes(Networking)
* Seguridad
* Aplicaciones para Dispositivos móviles
* Aplicaciones de tipo desktop
* Código controlador de máquinas y/o electrodomésticos
* Otros ámbitos

1. El Kit de desarrollo o JDK Tiene dos implementacioes principales, la de Oracle, llamada JDK y la de la comunidad Libre, el OpenJDK, que es el que vamos a utilizar
2. Como Java se puede implementar en diferentes ámbitos de la tecnología, se ofrece en difrerentes versiones, cada una con su propio SDK:...

* **JSE O Java Standard Edition:..**.

Ésta versión de Java se utiliza básicamente para crear aplicaciones del tipo Desktop, como aplicaio independientes y aplicaciones distribuídas.

Nuestro aprendizaje inicial estará concentrado en ésta versión de Java

* **JEE o Java Enterprise Edition:...**

Es la versión de Nivel Empresarial para crear aplicaciones a gran escala(Aplicaciones Empresariales de alto nivel y manejo crítico de datos, Aplicaciones Web, aplicaciones de comercio electrónico)

* **JME o Java Micro Edition:...**

Es la versión utilizada al crear aplicaciones para dispositivos móviles, electrodomésticos, cámaras de seguridad, routers, otros dispositivos electrónicos de menor capacidad de procesamiento y/o almacenamiento.

1. Al igual que otros lenguajes de PRogramación, Java puede interactuar con otros lenguajes de Programación, mediante API’s, ya sea para consumir recursos o para ofrecerlos
2. **PENDIENTE**

**Funcionamiento general de una aplicación Java**

A grandes rasgos, podemos representar el proceso de creación y ejecución de un programa Java de la siguiente manera:...

**Archivos .java**

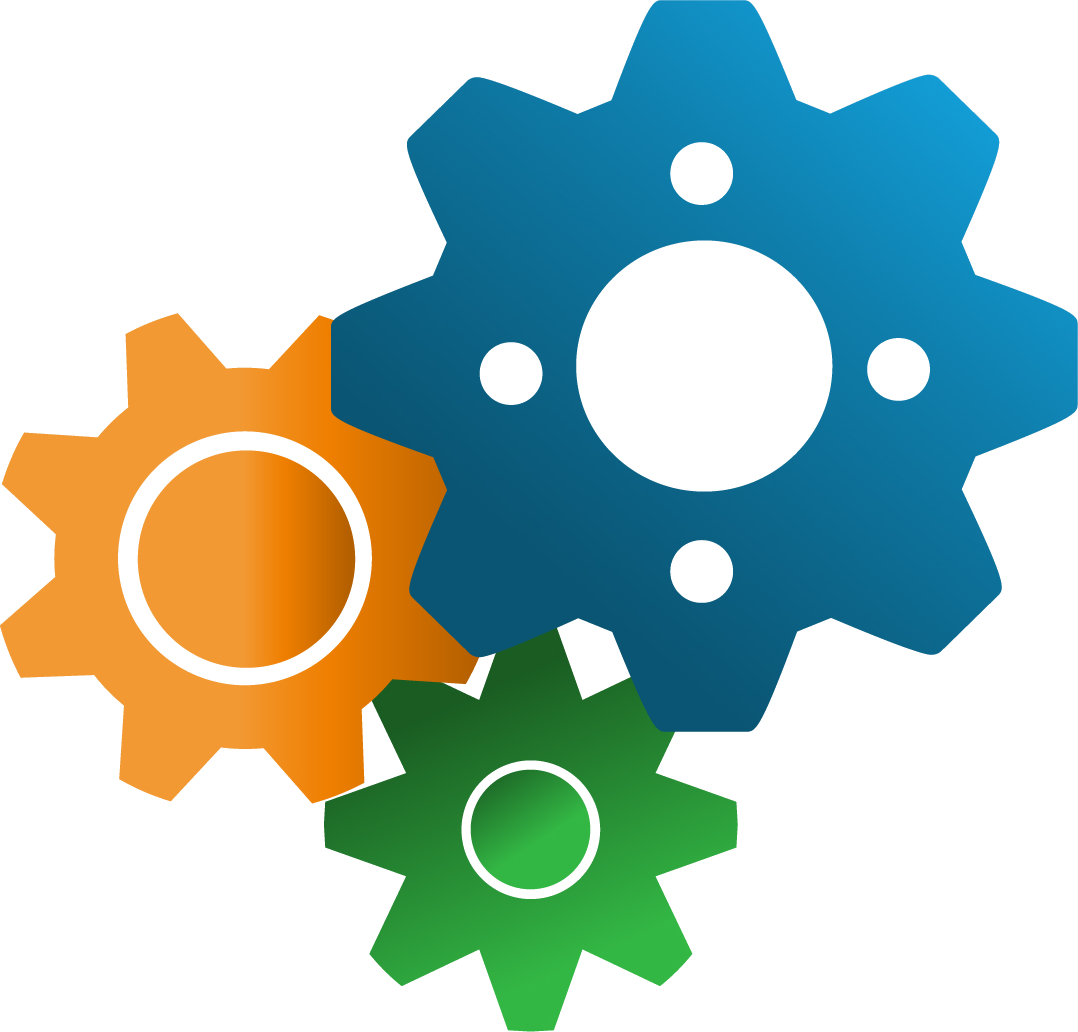
**Saved as**

**Class2**

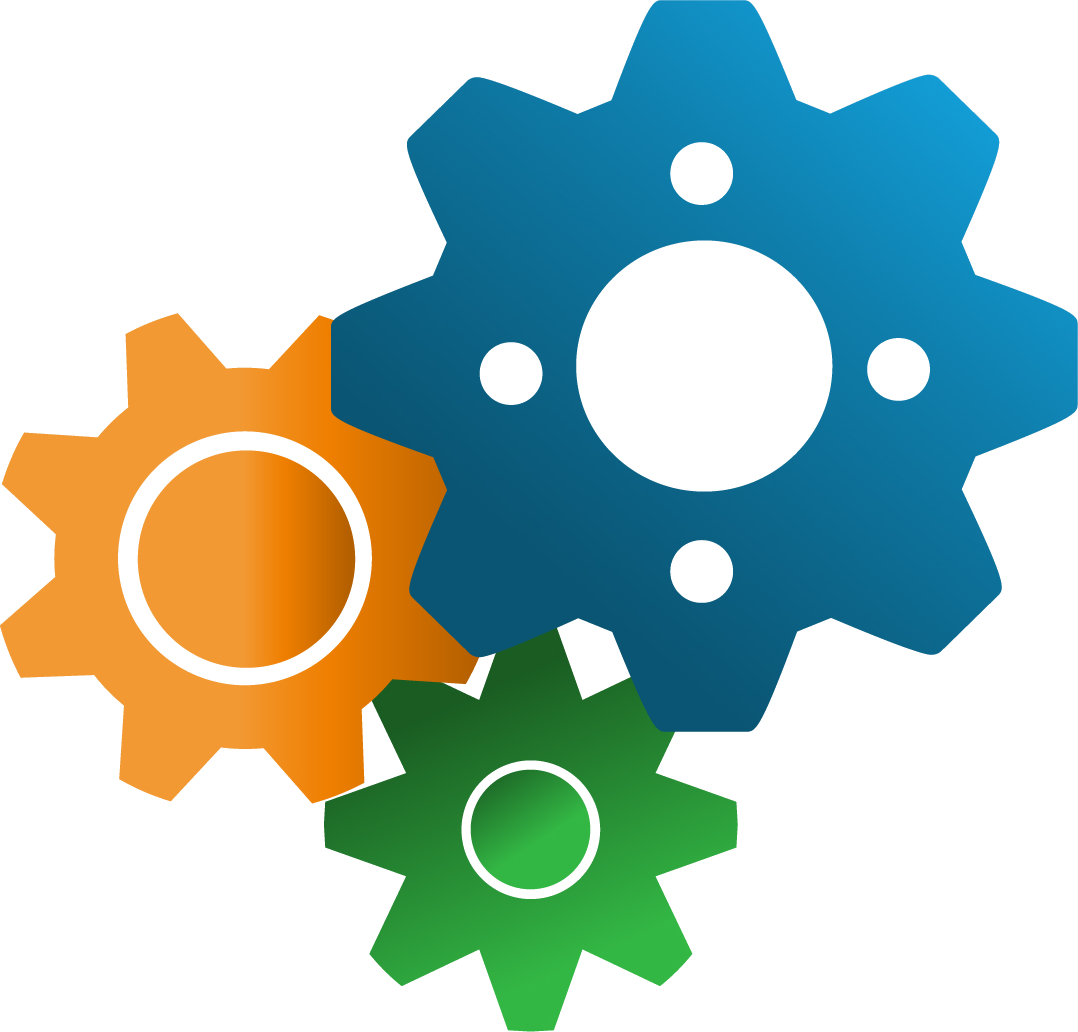
**Class1**

**Nuestro código implementado, organizado y conectado mediante clases**

**Class3**

****

**JavaCompiler...**

****

**Class1**

**JVM**

**Class2**

**Read with**

**Archivos .class**

**Class3**

**(ByteCode)**

**Ejecución**

**Output or Results**

1. Una de las cualidades de Java es una implementación de un Garbage Collector(recolector de basura) que se encarga de asignación de datos en la memoria RAM de la computadora, como también se encarga de eliminar lod datos que ya no están siendo utilizados durante la ejecución y/o que ya no es necesario mantenerlos(liberación automática de memoria utilizada), de manera que el programador puede concentrar sus esfuerzos unicamente en la creación de datos primitivos(variables comunes) y objetos, como de las operaciones entre estos. La gestión de memoria corre por cuenta de la JVM.
2. Java es Case-sensitive y esto debemos tenerlo en cuenta cuando vamos a utilizar variables u objetos que ya hemos implementado
3. Recordemos que las Funciones y procedimientos en la Programación Orientada a Objetos pasan a llamarse “métodos”

Aunque seguimos llamandolos Funciones cuando ejecutan el dódigo que contienen y retornan un valor, y procedimientos cuando ejecutan su código pero no retornan valor alguno.

**Picando Código:...**

Para empezar a escribir o “picar” código de Java. Es importante saber sobre que versión de la plataforma Java estamos trabajando. Para ello podemos ejecutar en nuestra terminal, dos comandos que nos muestran, por un lado la versión del JDK utilizado y por otro lado, la versión del compilador JavaC, implementados en nuestro sistema operativo:...

* Java --version

openjdk version "11" 2018-09-25

OpenJDK Runtime Environment (build 11+13-Debian-2)

OpenJDK 64-Bit Server VM (build 11+13-Debian-2, mixed mode, sharing)

* Javac --version

javac 11

Podemos ver que tenémos en nuestra computadora el OpenJDK de 64 Bit y el compilador JavaC, ambos en su versión 11

**Nuestro primer programa con Netbeans:...** (video 11)

* Creamos un nuevo proyecto File, New Project, Java, Java Application
* Botón Next
* Ingresamos el nombre de nuestro proyecto(java\_one) y su ubicación. Como es la creación de nuestro proyecto, habilitamos la opción “create Main Class” e ingresamos en esa casilla “Main”

Main Class es el archivo de arranque de los proyectos Java

* Botón Finish
* Vemos entonces que se genera un proyecto básico llamado “java\_one” con un subdirectorio “Source Packages” y dentro de éste, un paquete de código que, asu vez contiene un archivo “Main.java” en el que vamos a implementar nuestro primer código.
* Podemos ver que el archivo ya tiene generada la clase Pública Main y dentro de ésta, está el metodo estático main, que es donde vamos a escribir nuestro primer código:...

public static void main(String[] args)

{

System.out.println(" Hello, I am demonscript!!!:... ");

}

* Presionamos el botón verde “Run Project” o la tecla F6, con lo que se inicia el proceso de verificación, compilación y posterior ejecucuón de nuestro programa. Resultado que puedo ver en el mismo Netbeans

**TRICK:...**

**Algo MUYYYYY!!! curioso que apenas descrubro es que como ya tenía instalado el sistema de control de versiones GIT. Puedo ver en el mismo Netbeans el Historial de cambios de mi archivo:...**

* + **Click derecho sobre el archivo**
  + **Opción “History”**
  + **“Show History”**

**Esto, desde luego que es MAGNIFICO!!! :)**

**Variables en Java:...**

Como en otros lenguajes, los datos en Java se almacenan en Variables y constantes, las variables existen de manera temporal en un espacio de memoria asignado automaticamente. Las variables son de tipos defeinidos y podemos clasificar esos tipos en:

* Tipos de Datos primitivos
* Tipos de datos Object (referencia a Objetos)

**Tipos de Datos Primitivos:...**

* **Enteros (int) Que a su vez se clasifican en:...**
  + byte de 8 bits, por defecto su valor es 0
  + Short 16 bits, por defecto su valor es 0
  + int de 32 bit, por defecto su valor es 0
* **Tipos char, que son de un solo caracter(lerta, numero u otro caracter):..**
  + char 16 bit, que utiliza el código unicode para almacenar valores, por defecto su valor es \u0000
* **Long, que es un entero de mayor embergadura que un int:...**
  + long de 64, por defecto su valor es 0
* **Floantes (float) que a su vez se clasifican en:...**
  + Float de 32 bit, por defecto su valor es 0.0
  + Double de 64 bit, por defecto su valor es 0.0
* **Boolean(bool), tipo de dato que almacena true o false, por defecto su valor es false**

**Tipos de datos Object:..**

* Clases (class)
* Interfaces (interface)
* Arreglos (Array)
* String es un tipo especial de Object para manipular Cadenas:...



En la tabla anterior se muestra la escritura para los caracteres de escape que vamos a utilizar para presentar mensajes en aplicaciones de Consola.

**NOTA**:

Praticando con implementaciones de métodos, me doy cuenta de que hay dos tipos generales de métodos, los etáticos y los normales. Ladiferencia radica en que para acceder a un método estático de una clase, no necesito crear instancias - objetos de la clase, los puedo llamar escribiendo directamente el nombre de la clase y a continuación invocar el método:..

Nombre\_Clase.nombre\_método\_static();

Esta particularidad genera inseguridad al trabajar con Threads.

Mientras que para acceder a los otros metodos , tengo que crear un objeto de la clase que los contiene:..

Nombre\_Clase Objeto = New Nombre\_Clase;

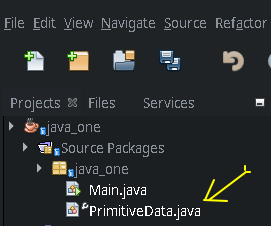
Objeto.nombre\_metodo();

Ahora, los metodos estáticos tienen la limitante de no poder acceder a los atributos y funciones, salvo que estos también sean declarados como static.

**Práctica 1 de datos primitivos:...**

En esta práctica jugamos con POO, pero lo importante de esto es la implementación y presentación en pantalla de los “datos primitivos”.

En el proyecto “Java\_one” que tenemos creado en Netbeans, Declaré en una clase independiente, en el mismo paquete donde está la clase Main:

 ...Class PrimitiveData

**Código:...**

public class PrimitiveData

{

//Vamos a implementar los tipos de datos básicos de Java:...

private int return\_byte;

private short return\_short;

private int return\_int;

private char return\_char;

private long return\_long;

private int response;

//Recibimos un string que tendrá un opción especifica

public void primitive\_data(String option)

{

switch(option)

{

case "byte":

return\_byte = 2;

response = 1;

break;

case "char":

return\_char = 'E';

response = 2;

break;

case "short":

return\_short = 10;

response = 3;

break;

case "int":

return\_int = 400;

response = 4;

break;

case "long":

return\_long = 56556674;

default:

response = 0;

break;

}

}

//Retornamos un valor tipo int, de acuerdo a la opción enviada con anterioridad

public int return\_primitive\_data()

{

if(response == 1)

{

return return\_byte;

}

else if(response == 4)

{

return return\_int;

}

else if(response == 3)

{

return return\_short;

}

else

{

return 0;

}

}

//Retornamos un valor tipo long:...

public long return\_primitive\_data\_long()

{

return return\_long;

}

//Retornamos un valor Tipo char:...

public char return\_primitive\_data\_char()

{

if(response == 2)

{

return return\_char;

}

else

{

return '0';

}

}

} //Fin del cuerpo de la clase PrimitiveData

Ahora, desde la clase Main, vamos a manipular el contenido de la clase PrimitiveData, creando una instancia de dicha clase

**Código para la Clase Main:..**

public class Main

{

public static void main(String[] args)

{

// TODO code application logic here

System.out.println(" Hello, I am demonscript!!!:... ");

//Creamos una instancia de la Clase PrimitiveData

PrimitiveData datos\_primitivos = new PrimitiveData();

//llamamos al método de los enteros similares:..

datos\_primitivos.primitive\_data("no");

System.out.println("El valor del dato tipo byte por defecto es "+datos\_primitivos.return\_primitive\_data());

datos\_primitivos.primitive\_data("byte");

System.out.println("Ahora El dato byte es "+datos\_primitivos.return\_primitive\_data());

System.out.println("");

datos\_primitivos.primitive\_data("no");

System.out.println("El valor del dato tipo short por defecto es "+datos\_primitivos.return\_primitive\_data());

datos\_primitivos.primitive\_data("short");

System.out.println("Ahora el dato tipo short es "+datos\_primitivos.return\_primitive\_data());

System.out.println("");

datos\_primitivos.primitive\_data("no");

System.out.println("El valor del dato tipo int por defecto es "+datos\_primitivos.return\_primitive\_data());

datos\_primitivos.primitive\_data("int");

System.out.println("Ahora el dato tipo int es "+datos\_primitivos.return\_primitive\_data());

System.out.println("");

/\*llamamos al método de los datos tipo char porque siutilizamos al método

\*de enteros similares nos devuelve un número que representa a la letra E

\*/

System.out.println("El valor del dato tipo char por defecto es "+datos\_primitivos.return\_primitive\_data\_char());

datos\_primitivos.primitive\_data("char");

System.out.println("Ahora el dato tipo char es "+datos\_primitivos.return\_primitive\_data\_char());

System.out.println("");

/\*llamamos al método de los datos tipo long porque int no soporta a long,

\*que es mucho mas grande porque int es 32 bit mientras que long es 64 bit

\*/

datos\_primitivos.primitive\_data("no");

System.out.println("El valor del dato tipo long por defecto es "+datos\_primitivos.return\_primitive\_data());

datos\_primitivos.primitive\_data("long");

System.out.println("Ahora el dato tipo int es "+datos\_primitivos.return\_primitive\_data\_long());

System.out.println("");

}

} //Fin del cuerpo de la clase Main