**Seccion 6- Java Basics for RestAssured Automation**

En esta seccion se usaran todos los archivos de las clases

**38. Setting up Java on Windows,Data Types & Operators in Java**

* Instalacion java
* Descarga e instalacion eclpse
* Variables
* Tipos de datos
* Operadores en java

**39. Java Control Statements**

* Selections statement:
  + If….else
  + Switch…case
* Iteration Statements:
* While
* Do…while
* For
* Jump Statements
* Break
* Continue

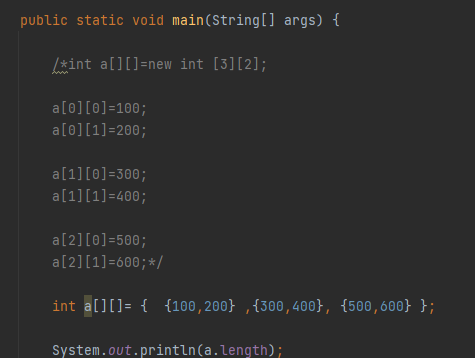
**40. Java Arrays & Strings**

* Arreglo single dimensional int a = new int [5]

Texto

Descripción generada automáticamente

* Arreglo de 2 dimensiones



3 filas 2 columnas

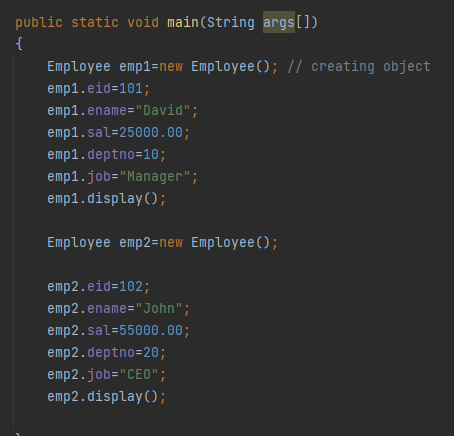
**41. Java OOPS Concepts, Class, Object, Methods & Constructors**

* OPP – Object oriented programming – programación orientada a objetos
* Se enfoca en clases, objetos y métodos
* Una clase es una colección de variables y métodos

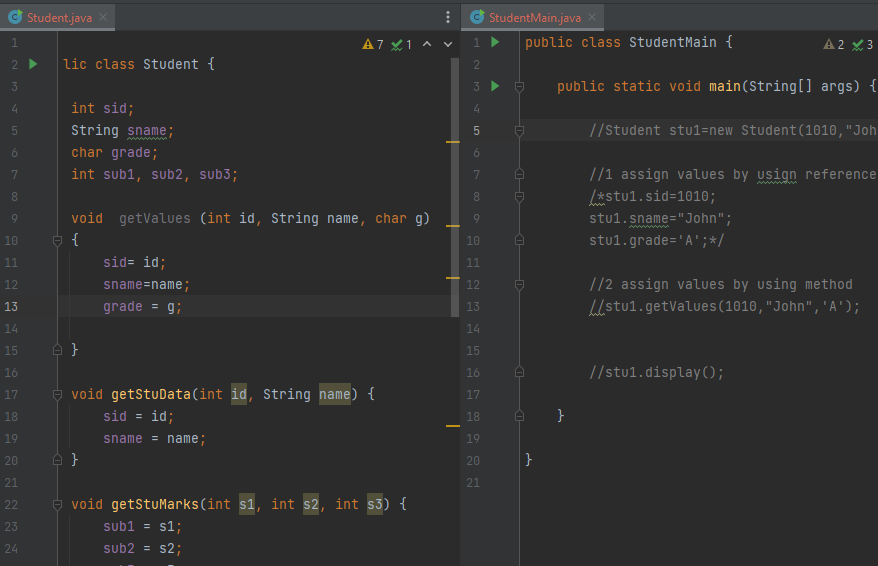
Por ejemplo se tiene la clase animales. Dentro de ella hay perros, gatos, etc.. Todos estos son objetos

Los métodos hacen referencia a el comportamiento

Los atributos son las variables



* En la documentación se tiene la clase Employee
* Los ejemplos en la clase EmployeeMain
* Objeto es una instancia de la clase . Se pueden crear la cantidad que se quiera de objetos
* Un metodo es un bloque de código que contiene la lógica.
* Main() methos puede estar presente en la misma clase
* Main() method puede estar presente en clases separadas
* Este es el caso de las clases Employee y employeeMain, la segunda llama desde la primera
* Para hacer esto ambas clases deben estar en el mismo proyecto, para poder llamar la una desde la otra.
* Variables de clase:
* Hay 3 formas de asignar valores una variable:
* Referenciando la variable
* Usando un método
* Usando un constructor
* En este caso se usará la clase Student y StudentMain



* Constructor:
* Es un tipo especial de método
* Es usado para inicializar las variables de clase
* El nombre del constructor debe ser igual al nombre de la clase
* El contructor no retornará valores
* El contructor es invocado en el momento en que se crea el objeto

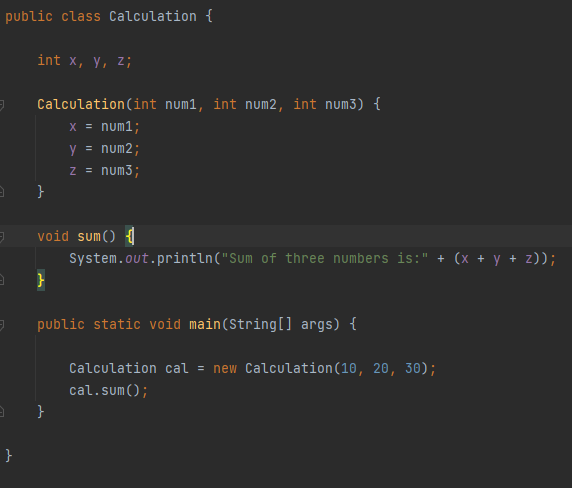
Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

* Métodos – Bloque de código
* Caso 1: no toma parámetros y no retorna valor
* Caso 2: no toma parámetros pero retorna valor
* Caso 3: toma parámetros pero no retorna valor
* Caso 4: Toma parámetros y retorna valor
* Se verá en la clase Calculation



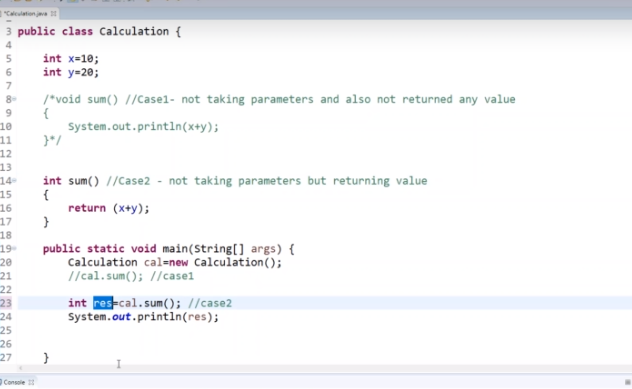


Imagen que contiene Texto

Descripción generada automáticamente

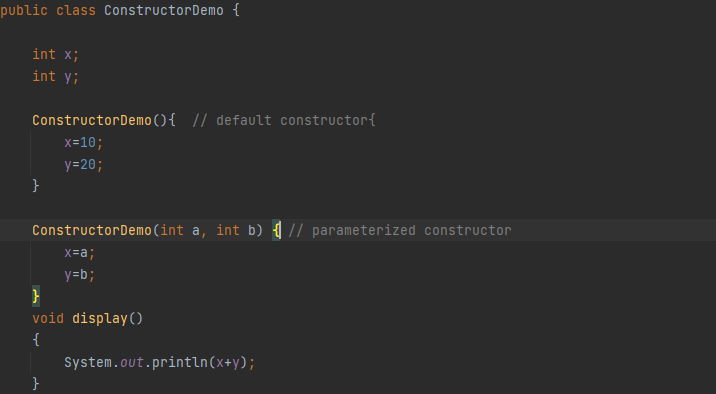
Texto

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* Tipos de constuctores:
* Constructor por defecto
* Constructor parametrizado
* Se verán en la clase ConstfuctorDemo



Texto

Descripción generada automáticamente

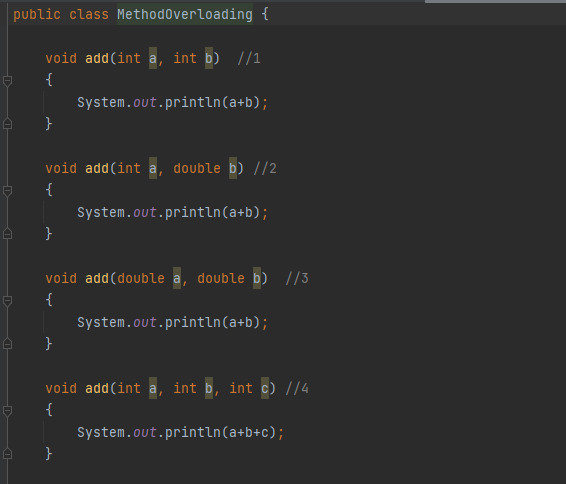
**42. Java Overloading, this Keyword & static keyword:**

1. Overloading
2. This keywordS
3. Static variables and methods

* Si se crean varios métodos estos se alojaron en distintas espacios de memoria
* Si se crea un solo método se va solo a un espacio de memoria, esto es lo que se llama overloading.
* Se crea uno que pueda hacer varias cosas
* Los métodos y los constructores se pueden Overload
* Overloading: Crear métodos/construuctores con el mismo nombre con diferentes parámetros.
* Ventajas: ahorra memoria, ahorra tiempo
* Para el overloading existen unas reglas:

1. Numero de parámetros
2. Tipo de dato del parámetro

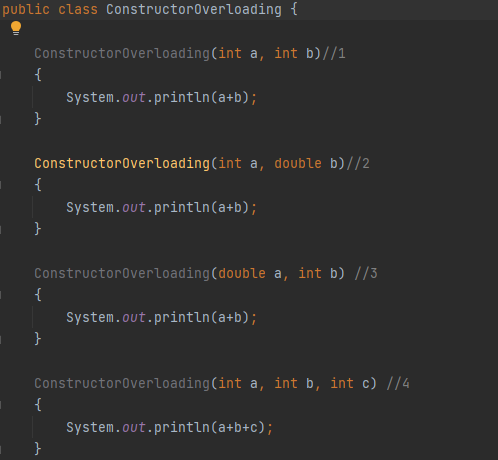
* El ejemplo de los métodos se verá en MethodOverloading
* Para el overloading de métodos, los métodos pueden tener cualquier nombre deseado y este se puede repetir.



Texto

Descripción generada automáticamente con confianza baja

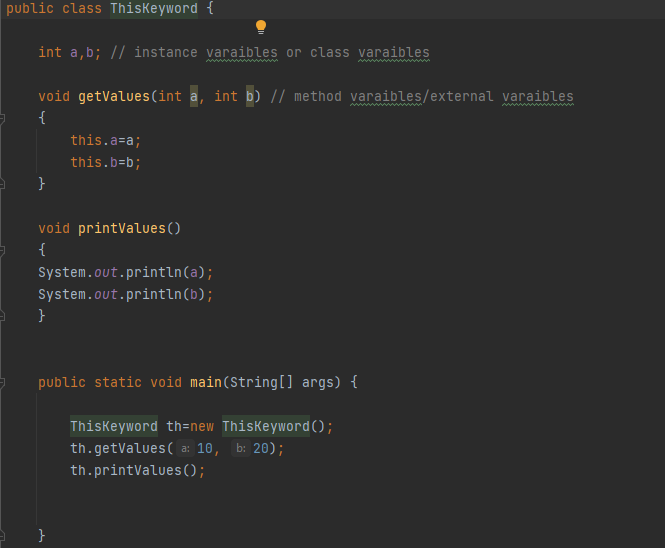
* Para el overloading de contructor, el nombre de los métodos debe ser el mismo de la clase



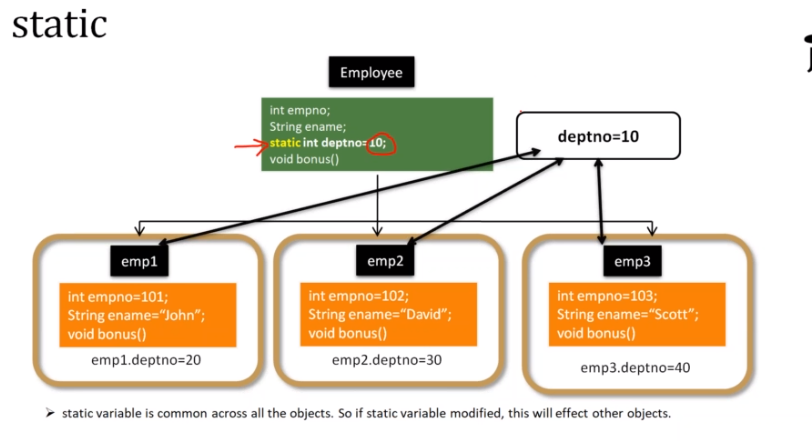
Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

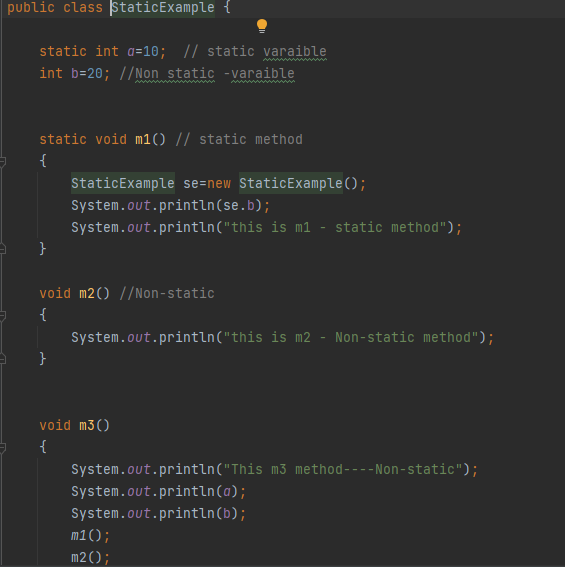
* **This Keyword:** Se cuando se tienen variables de clase y variables de método con el mismo nombre. Se usa para que el método las diferencia**.** Cuando se usa el this , apunta a las variables de clase



* **Static:** aplicable para variables y métodos
* Si no es estático, todos los objetos ocupan un cierto espacio en memoria, es decir, si se tienen 100 personas, ocupará por ejemplo 100 espacios de memoria para un mismo valor

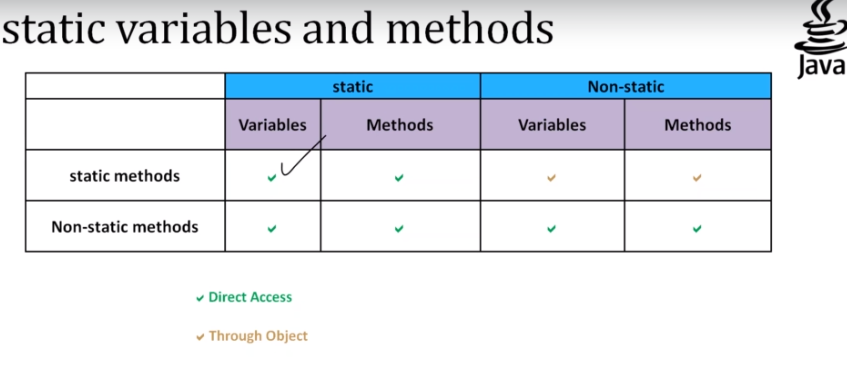


* Con Static, el departamento ocupará solo una locación es memoria
* Se verá en la clase StaticExample
* Los métodos Staticos solo pueden acceder cosas estáticas directamente
* Los métodos staticos pueden acceder a las cosas no estáticas pero solo a través de objetos
* Métodos no staticos pueden acceder tanto a cosas estatisas y no estaticas



Texto

Descripción generada automáticamente



**43. Java Inheritance,Overriding,super & final keywords:**

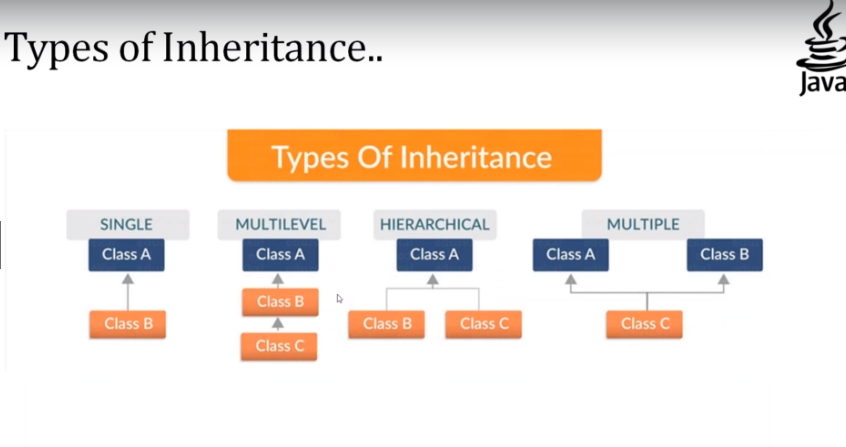
1. Java inheritance
2. Mmethod overriding
3. Super keyword
4. Final Keywod

* **Java inheritance:** Es una extensión de una clase que contiene ya variables y métodos. Entonces en una clase B puedo usar las mismas variables y métodos de la primera además de crear otros
* **Inheritance:** obtenemos variables y métodos de otra clase

Parent/ base class /supper class -- child class/sub class / derived class

* Hay 5 tipos de inheritance;:

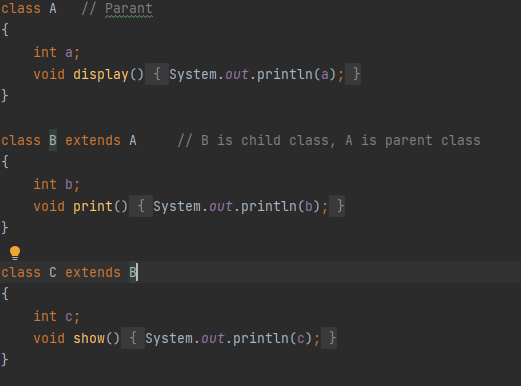
1. Single
2. Multiple level
3. Hierarchical
4. Multiple
5. Hybird
6. Single: Se va de A a B
7. Multiple lvel: Se tienen varioa niveles, B sale de A y C sale de B
8. Hierarchical: Varias que viene desde A, es decir, B C y D salen de A
9. Multiple: En este caso C viene de A y de B, al mismo tiempo
10. Hibrido: Es la combinación de Multiple y hierarchical. Es decir: D sale de C y de B, Y a su vez, B y C salen de A.

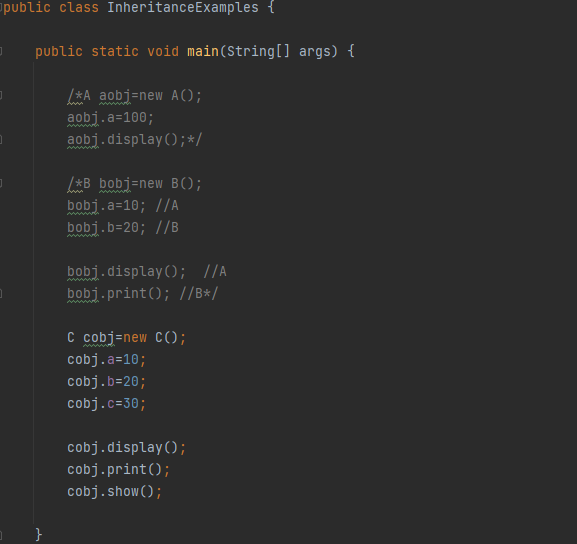


Texto, Aplicación

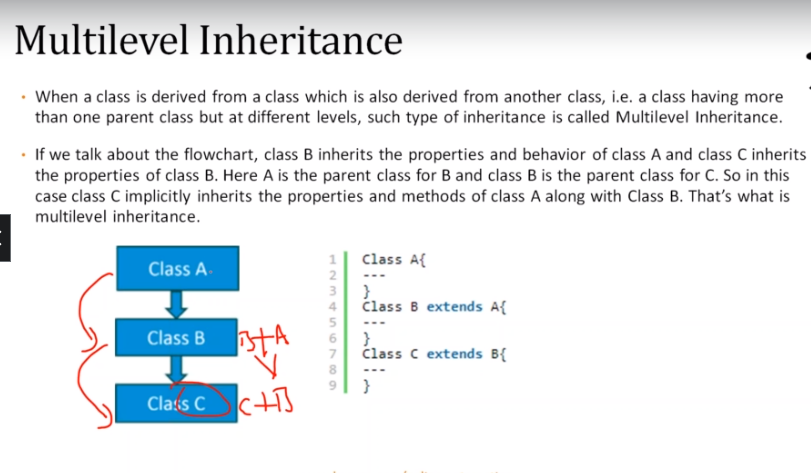
Descripción generada automáticamente

* Se verá en la clase InheritanceExample

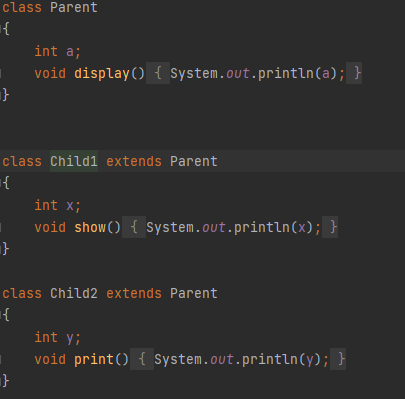




* Con aobj solo se puede acceder a A
* Con bojb Se pouede acceder a lo de A y B
* Con Cobj se puede acceder a todo lo de A B y C



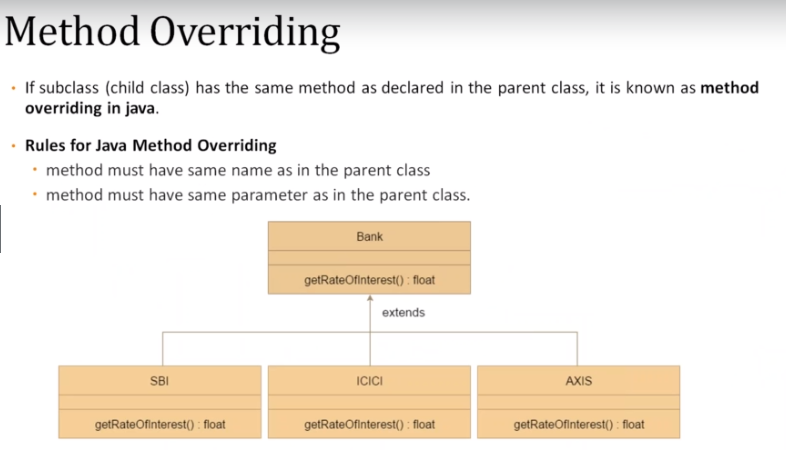
* Para hierarchical se crea una nueva clase Hierarchical Inheritance



Texto

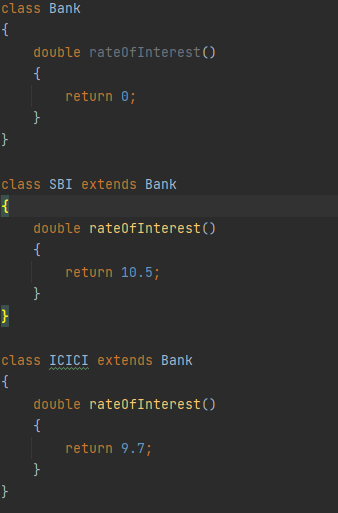
Descripción generada automáticamente

* Las otras 2 implemenataionces se verán en la próxima clase
* **Method Overriding:** Es cuando un mismo método m1 se define en A , Luevo una clase B extiende de A, y en B se crea un nuevo método m1. Este nuevo método debe recibir los mismos parámetros, pero, el body si puede ser diferente.
* **Diferencias overloading y overriding:**
* En overloading se debe cambiar la definición, numero de parámetros y tipo. En overriding solo se cambia el body
* Overloading puede ser usado in inheritance. Overriding debe al menos tener 2 clases, una padre hijo



* Se revisará en la clase Method overrriding

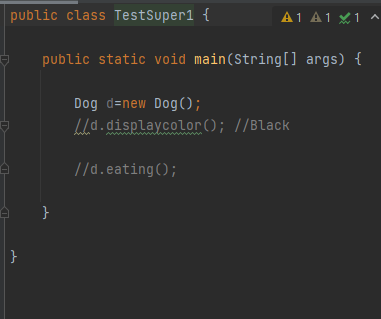
Texto

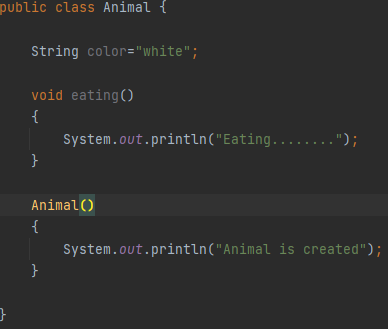
Descripción generada automáticamente

* **Super Keyword:** Solo se puede usar en inheritance**,** puede ser usado en:

1. Super puede ser usado para referenciar la variable de instancia de un Parente inmediato
2. Super puede ser usado para invocar una clase/método de un pariente inmediato
3. Super () puede ser usado para invocar el contuctor de clase de un pariente inmediato

* Esto se verá en la clase Animal y Test Super

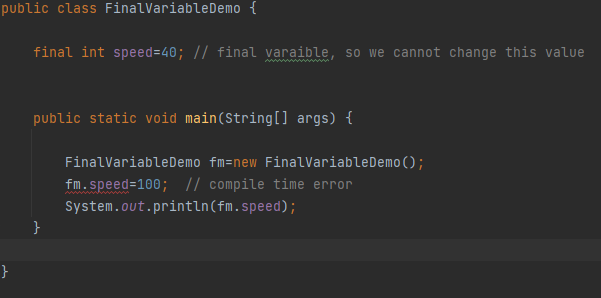
Texto

Descripción generada automáticamente

* **Final Keyword:** Se usa para que ya no se pueda cambiar el valor. Hay 3 formas de usarlo:

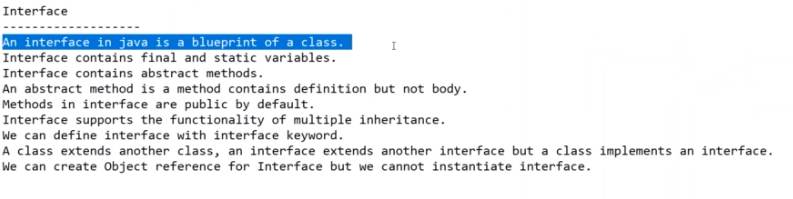
1. Variable: no se puede cambiar el valor de la variable
2. Método: el método no puede ser sobreescrito por los hijos
3. Clase: la clase no se puede extender de otra

* Esto se verá en la clase FinalVariableDemo, FinalMethodDemo y FinalClassMethod



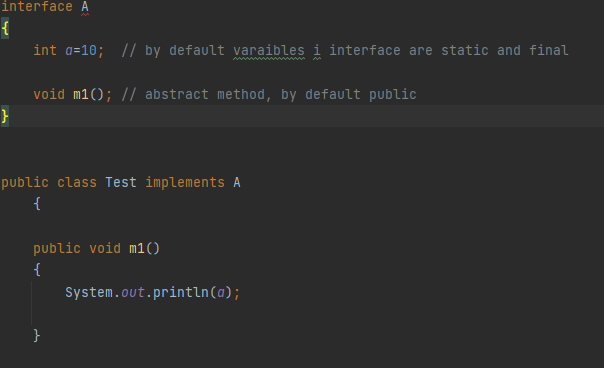
**44. Java Interfaces,Java Packages & access Modifiers:**

* Java interfaces
* Java packages
* Access modifiers
* **Java interface:**

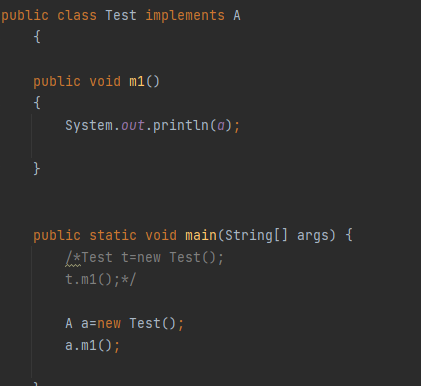


* En inheritance, de una clase a otra clase se usa extend, de una interface a otra interface también se usa extend, pero, de ina interface a una clase se usa la palabra ‘implement’
* De una clase a una interface, no se puede, no es posible
* Esto se puede usar por ejemplo cuando se necesita automatizar pruebas en diferentes navegadores, una clase puede ser pasa Chrome en la cual se contienen los drivers, en otra los de Safari, etc…
* Revisaremos esto en la clase Test

Texto

Descripción generada automáticamente

* En la clase Test se debe implementar el método m1 que está dentro del interface
* En el método main se accede al método
* No se puede instanciar una interface como = new interface(). Pero si se puede instanciar desde la clase



* **Multiple inheritance:** Se tiene múltiples padres para un solo hijo.
* Los padres se crearan como unas interfaces y el hijo como una clase que implementa las interfaces. Se usa la palabra reservada ‘implement’
* No se pueden usar clases como padres porque esto no permite heredar múltiples clases a un solo hijo. Por esto es que se usan las interfaces. Una clase solo puede heredar a una clase al mismo tiempo.
* Se revisará en la clase MultipleInheritance

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

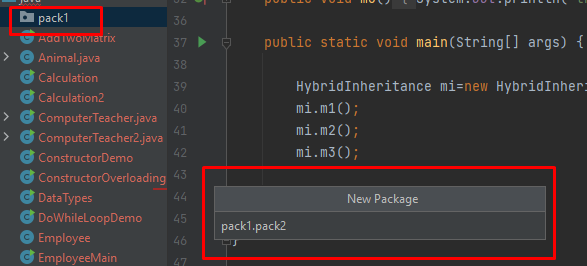
* Hybrid inheritance: B y C son hijos de A, y D es hijo al mismo tiempo de B y C. La primera paret se puede hacer con clases, la segunda solo con interfaces
* Se revisará en la clase HybridInheritance

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

* **Java Packages:**  Es un tipo de folder que contiene algunas clases , en la cuales hay alguna funcionalidad.
* Si quisiéramos una clase para usarla en otra clase o algo parecido, nos tendríamos que referir a ella por el paquete . Tendríamos que importarlo con el comando import …..
* Cuando se quiere crear un paquete dentro de otro, se le da click y luego crear paquete pero el nombre debe ser por ejemplo pack1.pack2

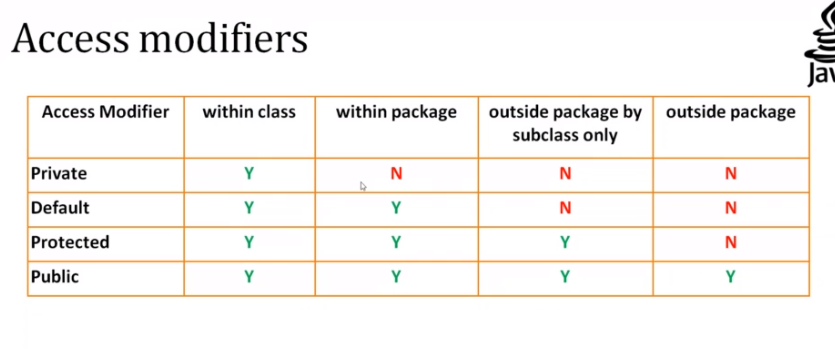


* Hay 2 tipos de packages:

1. User defined packages
2. Build in packages

* **Access Modifiers:** Define el scope de las variables y los métodos

1. Public: Se pude acceder desde cualquier punto sin necesidad de extender
2. Protected: solo puede ser usado en las inheritance pero puede ser usado desde otros paquetes. La clase se debe extender y también se debe importar el package
3. Default: Si no se usa algún modificador se queda como default y este solo se puede usar dentro del mismo package
4. Private: Solo se puede acceder dentro de la misma clase



**45. Exception Handling in Java:**

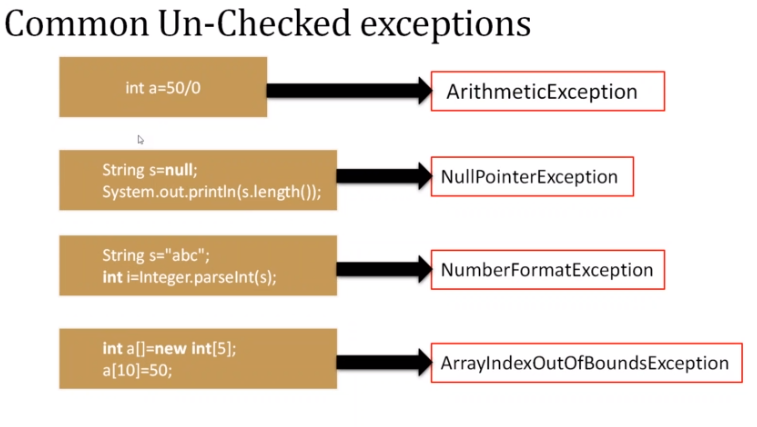
* Primero se debe conocer la diferencia entre un compilador y un interprete. El compilador corre el código línea por línea y cuando encuentra un error se sale y lo muestra. Por el contrario un interprete lo corre todo y al final muestra el numero de errores
* **Exception:** Es una condición anormal, en java, una exception es un evento que interrumpe el Flow normal del programa
* **Tipos de excepciones:**

1. Checked: Las excepciones pueden ser detectadas por el compilador. En estas hay algunos ejemplos

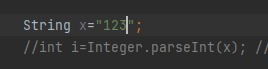
* InterrumptedException
* IOException
* FileNotFoundException….

1. Un -checked: No pueden ser identificados por el compilador y suceden en tiempos de corrida del programa. Ejemplos

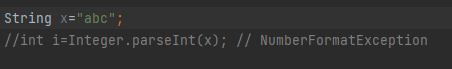
* ArithmeticException
* NullPointerException
* NumberformatException
* ArrayIndexOutOfBoundsException
* Primero revisaremos las un checked en el Exceptionexample



* En Java se puden convetir por ejemplo un String en un int de la siguiente manera



* Los valores están almacenados en en String y con el Integer.parseint los concierte en un int. El problema se daría cuando por ejempolo son unas letras que se quieren convertir en un int por ejemplo:

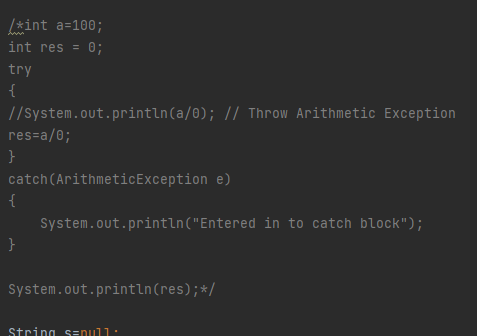


* **Handling expceptions:**  Se usa el try and catch

Texto

Descripción generada automáticamente con confianza media

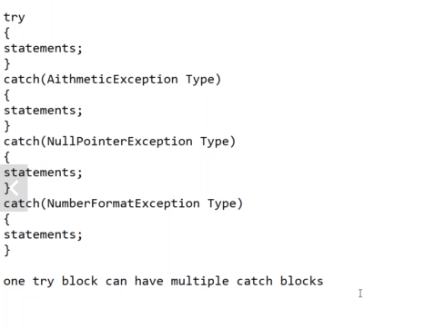
* Se incluye el tipo de exception y se dice que hacer cuando eso pase**.**



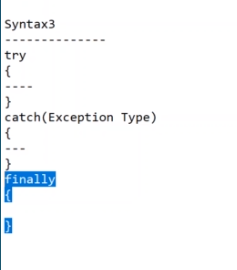
Interfaz de usuario gráfica, Texto

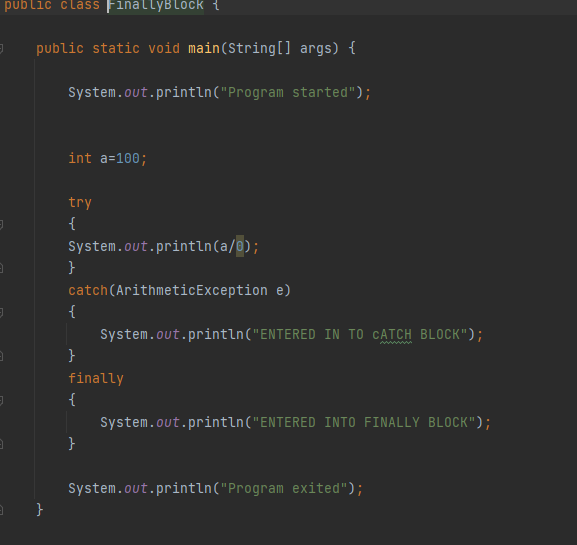
Descripción generada automáticamente

* Un solo bloque de TRy puede tener múltiples exceptions. Esto se puede usar cuando no se sabe que exception va a salir

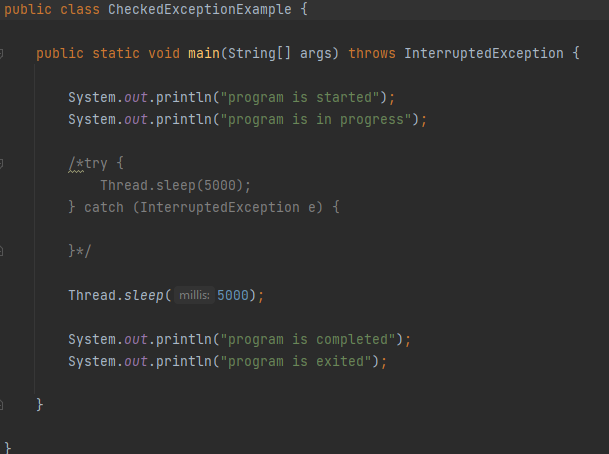


* Hay una sintaxis mas que añada un bloque llamado finally.
* Este bloque siempre se ejecuta. Es opcional pero siempre se ejecuta tanto si hay una excepción o no.
* En este bloque se pone normalmente por ejemplo el cerrar un archivo que se abrió en el try.
* Esto se revisará en la clase FinallyBlock

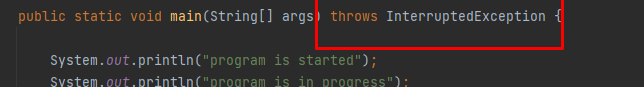




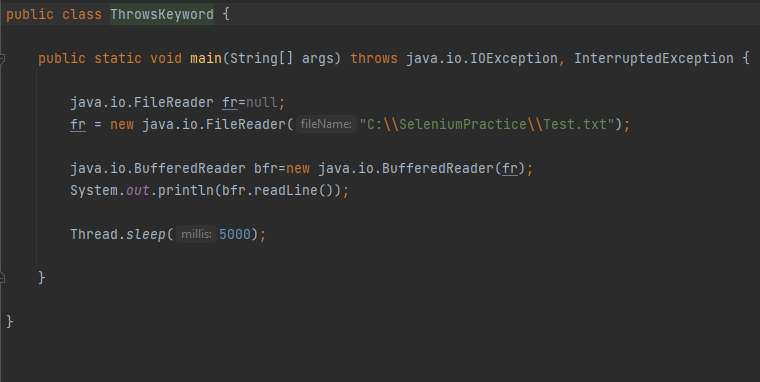
* Por ultimo veremos las Cecked exceptions en la clase CheckedExceptionsExample

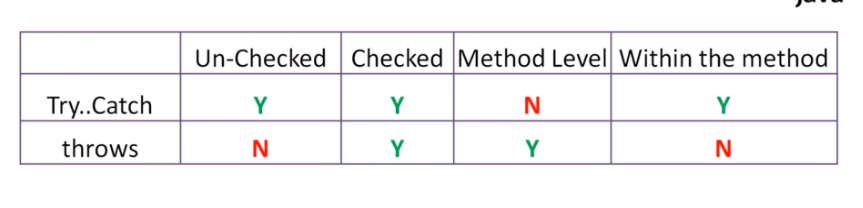


* Checked exceptions puede ser manjedas de 2 formas:
* Try and catch
* Throws keyword



* Se verá mejor en la clase ThrowsException



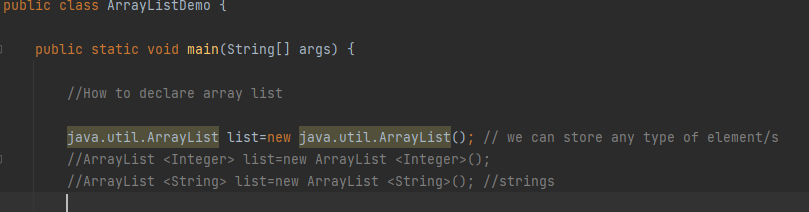


**46. Java ArrayList,HashMap & JDBC:**

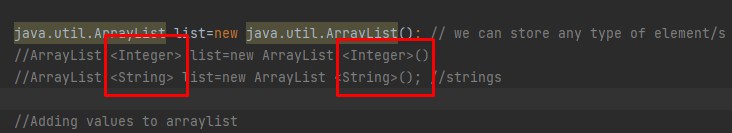
* ArrayList
* HashMap
* JDBC
* **Derived Data Types (Collections)**

Int a[] = new int a[5]

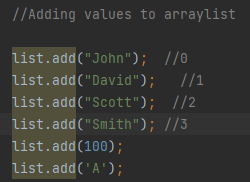
* Con este tipo de arreglos solo podemos guardar el numero de elementos con el que lo declaramos y por otro lado solo podemos almacencer el tipo de elementos que declaramos
* **ArrayList:**  Es un tipo de arreglo en el cual se pueden almacener cualquier numero de elementos y cualquier tipo
* **Operations in ArrayList:** 
  + How to declare
  + How to add elements/values
  + Find the size
  + Read values
  + Remove elements
* Se revisará en la clase ArrayListDemo



* Si se quiere almacenar un tipo en especifico de datos se debe crear el arraylist con <Tipo de dato>



* Para agregar valores



* Para imprimir el tamaño

Texto

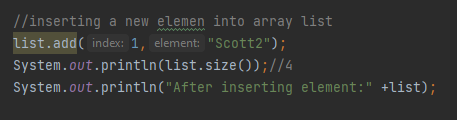
Descripción generada automáticamente

* Remover

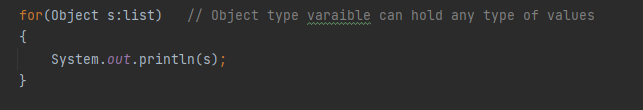
Texto

Descripción generada automáticamente

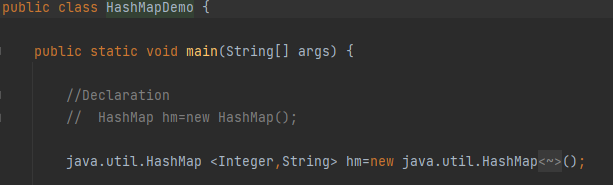
* Para insertas algún valor en el medio del arreglo . Se debe agregar el index y el valor



* Para leer los valores con un loop . Se llama un object porque esto puede almacenar cualquier tipo de dato.



* **Hashmap:** También es un tipo de arreglo pero en este se puede usar cualquier formato de datos. Este se usa con un key y un value, es decir un pair, un par de valores para poder referenciar.
* Los keys son únicos pero los valores pueden ser duplicados
* En el hashmap se puede:
* Como declarar un hashmap
* Añadir elementos
* Remove elements
* Read pair values
* Se revisará en la clase HashMapDemo
* Declaración: se declaran los tipos de valores que se ingresarán. Los kesy y valores pueden ser de cualquier tipo



* Para añadir parejas

Texto

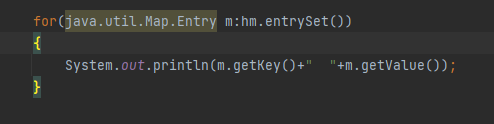
Descripción generada automáticamente

* Para remover

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

* Leer con un loop. Se usa el método entryset() el cual retorna la pareja completa. Si se quisiera obtener solo uno se tendría que usar m.getKey() o m.getValue().



* **JDBC – Java Database Connectivity**

Se usa SQL developer para conectar con la DB

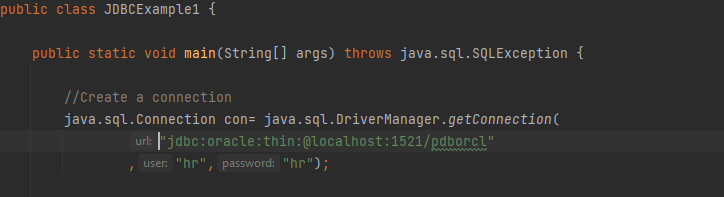
* Para usar JDBC se debe:

1. Descargar un driver de la DB (jar file)
2. Agregar el .jar al proyecto (se da click en el proyecto, properties y se adiciona el jar)
3. Ahora se puede escribir el programa

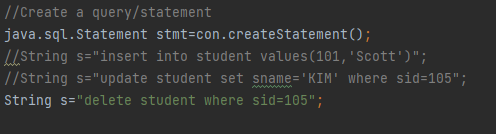
* Pasos para escribir el código:

1. Crear la conexión
2. Crear la Query o las sentencias
3. Ejecutar la Query a través de la conexion
4. Cerrar la conexión

* Se trabajará en JDBCExample1
* Crear la conexión:



* Crear la Query



* Ejecutar la Query



* Cerrar la conexión

Texto

Descripción generada automáticamente

* SE revisa la clase JDBCExample2

