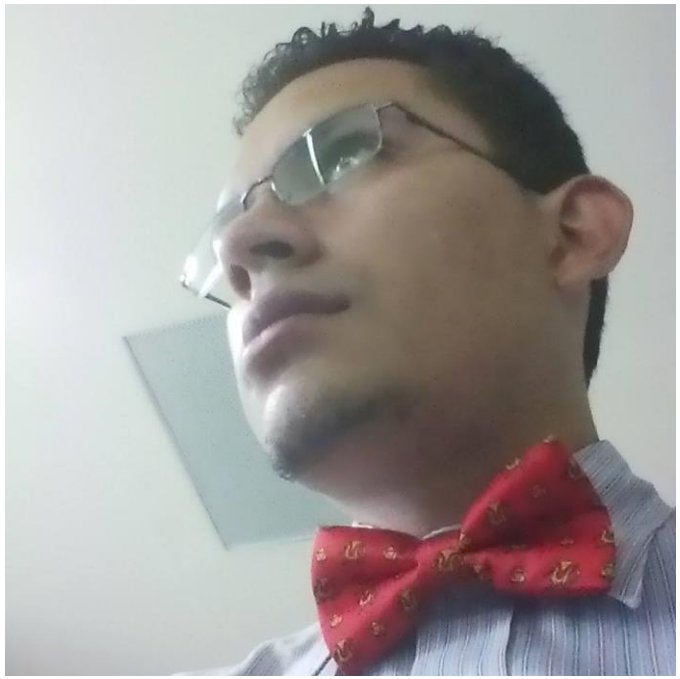
Java™

Buenas Prácticas de Programación

INVERSIONES INTELECTUALES

¿ Quien ?



Jorge Alejandro Cajas Mota

jac.mota@gmail.com
@cajasmota

Contenido

Temas

- ¿Porqué?
- Convenciones
  - Archivos
  - Paquetes
  - Clases e Interfaces
  - Variables
  - Métodos
- Buenas prácticas
  - Inicializar Variables
  - Alcance de las Variables
  - Ciclos
  - Excepciones
  - Strings

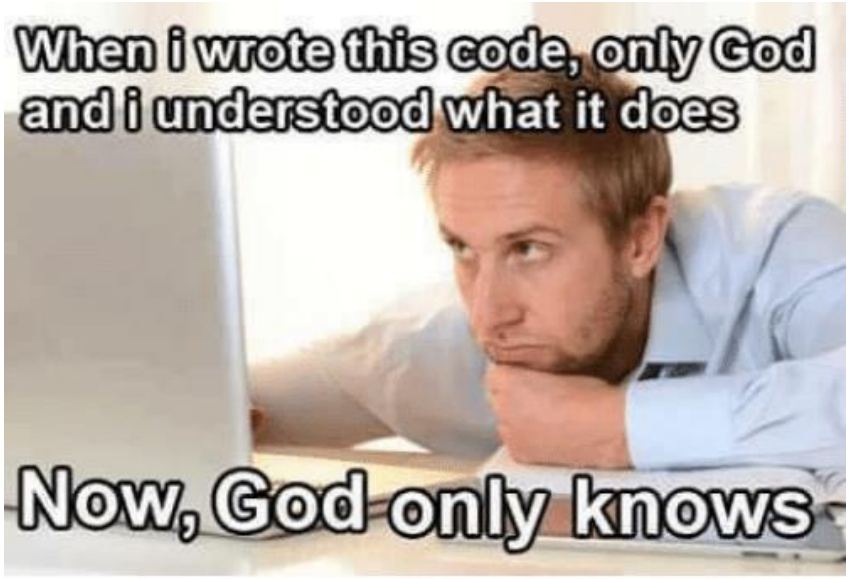
¿Porqué?

Lo modales hacen al hombre

Convenciones

Así todos nos entendemos…

- Son un conjunto de reglas que nos ***sugieren*** como organizar y escribir nuestro código.
- No son reglas obligatorias, pero facilitan la lectura y entendimiento del código fuente.
- Facilita el trabajo en equipo, la resolución de errores y la adición de nuevas funcionalidades



Convenciones

Archivos

- El código fuente del código java debe tener extensión *“**.java**”.*
- Los archivos de código fuente deben respetar la convención *“**PascalCase**”.*

MyClass.java	✓
myClass.java	X
myclass.java	X

Convenciones

Paquetes

- Los paquetes de código fuente deben respetar la convención "***lowercase".***
- El nombre del paquete debe ser descriptivo con el contenido o funcionalidad que engloba.

com.example.application	✓
Com.example.Application	X
com.example.a	X

Convenciones

Clases e Interfaces

- El nombre de una Clase o Interfaz debe respetar la convención "***PascalCase***".
- Cada archivo fuente de Java contiene un sola clase o interfaces publica.
- Cuando una clase o interfaces privada esta asociada con una clase publica, se puede poner el mismo archivo que la clase publica. Pero La clase publica debe ser la primera clase o interfaces en el archivo.

```
1. public class MyClass {
2.     private class MyPrivateClass {
3.
4.     }
```

Convenciones

Variables

- Los nombres de las variables deben ser cortos, pero descriptivos
- El nombre de las variables no debe comenzar con guiones bajos o signos de dólar.
- El nombre de una variable debe respetar la convención "***camelCase***"
- Si es una constante el nombre debe respetar la convención "***UPPERCASE***" y cada palabra se separa con un guion bajo

String name; ✓

final int ACCESS_CODE = 123; ✓

String Lastna; X

final int LastUsedAndAcceptedAccessCodeForDoor5 = 123; X

Convenciones

Métodos

- El nombre de método debe ser descriptivo.
- El nombre de un método debe respetar la convención "***camelCase***".

public void doSomething(); ✓
private int getResult(); ✓
public void DoSomething(); X

Buenas Prácticas

Porque todo puede ser mejor…

A diferencia de las convenciones, las buenas practicas no son reglas definidas que deberíamos de seguir, pero si son tips que nos ayudaran a escribir código más eficiente.

Buenas Prácticas

Inicializar Variables

Inicializar una variable (y más si es un objeto) es una tarea bastante costosa en la ejecución de nuestro código, por lo que debemos hacerlo hasta que sea necesario.

```
public class Paises {

    private List paises;

    public Paises(){

    }

    public List getPaises() {
        //se inicializa solo cuando es requerido
        if(null == paises) {
            paises = new ArrayList();
        }
        return paises;
    }
}
```

Buenas Prácticas

Alcance de las Variables

- Las variables locales pueden insertar mas bugs durante el copiado y pegado de código viejo.
- Minimizar el alcance de una variable local hace que el código sea mas legible, menos propenso a errores y mas mantenible.
- Debemos procurar declarar variables justo antes de ser usadas.
- Procurar inicializar una variable desde su declaración. Si eso no es posible asígnale el valor null.

Buenas Prácticas

Ciclos

Evitemos llamar funciones o métodos dentro de un ciclo si no es necesario.

```
public class Paises {

    private List paises;

    public List imprimirPaises() {

        for(int i = 0; i< paises.size(); i++)
        {
            System.out.println( paises.get(i) );
        }

    }
}
```

Buenas Prácticas

Ciclos

Evitemos llamar funciones o métodos dentro de un ciclo si no es necesario.

```
public class Paises {

    private List paises;

    public List imprimirPaises() {

        int tamaño = paises.size();
        for(int i = 0; i< tamaño ; i++)
        {
            System.out.println( paises.get(i) );
        }

    }
}
```

Buenas Prácticas

Excepciones

- No debemos ignorar las excepciones.
- Debemos procurar utilizar un solo bloque de Try y que el manejo de Excepciones debe ser desde la más específica a la más genérica.
- No debemos de hacer que un método lanze "***Exception***" ya que esto complica el manejo de errores.

Buenas Prácticas

Excepciones

```
public void doSomething() throws Exception {

    //Quierete un poco, no hagas esto....

}


public void doSomethingComplex(){
    try {
        /* code */
    }
    catch( FileNotFoundException ex){ /* code */ }
    catch( IOException           ex){ /* code */ }

}
```

Buenas Prácticas

Strings

- Los String son objetos inmutables y costosos de crear.
- Evitemos usar la concatenación de Strings con el operador "***+=***"
- Evitemos la concatenación de Strings en ciclos, usemos en su lugar ***StringBuffer*** o ***StringBuilder***

Buenas Prácticas

Strings

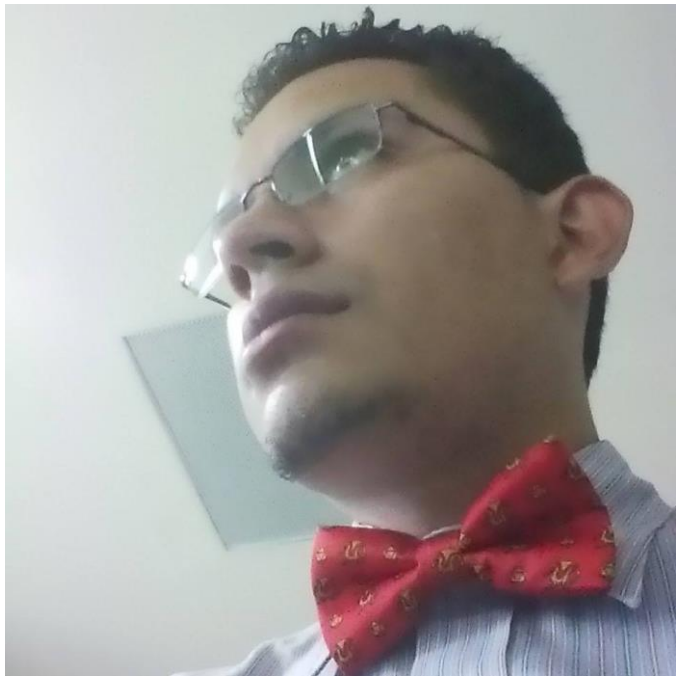
```
String numbers = "";
for(int i = 0; i<100; i++){
    numbers += " "+i;
}
System.out.println(numbers) //prints 1 2 3 4 ....
```

```
StringBuilder builder = new StringBuilder();
for(int i = 0; i<100; i++){
    builder.append(" ").append(i);
}
System.out.println( builder.toString() )  // prints 1 2 3 4 ....
```

Java™

¿ Preguntas ?

¡ Muchas Gracias !



Jorge Alejandro Cajas Mota

jac.mota@gmail.com

@cajasmota

INVERSIONES INTELECTUALES