TALENTO DIGITAL
INTELIGENCIA HUMANA

Talento Digital para Chile:

MÓDULO 1
PROGRAMACIÓN BÁSICA EN JAVA

UN PROYECTO DE:

Ministerio de Hacienda · Ministerio del Trabajo y Previsión Social · CORFO · sence · SOFOFA · OTIC SOFOFA Capital Humano · CPC · ACTI · BID · FCH Fundación Chile · kodea

DESARROLLADO POR:

MÓDULO 1 - PROGRAMACIÓN BÁSICA EN JAVA

1.2 EL ENTORNO JAVA PARA LA PROGRAMACIÓN

Semana 3 - Día 11

Objetivo de la jornada

- Entender y aplicar los distintos estándares, convenciones y estilos de codificación en el código utilizado para dar solución a un problema según requerimientos establecidos.

Estándares

En programación, seguir buenas prácticas, estándares y/o convenciones, beneficia de gran manera la lectura, modificación y mantenimiento de los sistemas. Alrededor de un 80% del coste del código de un programa va a su mantenimiento, además de que casi ningún software lo mantiene toda su vida el autor original por lo que se hace de vital importancia mantener un código ordenado y comentado, permitiendo entender código nuevo mucho más rápidamente y más a fondo. Para que funcionen estas buenas prácticas, cada persona que escribe software deben seguirlas al pie de la letra.

Sangría

Como norma general se establecen 4 caracteres como unidad de sangría. Los entornos de desarrollo integrado (IDE) más populares, tales como Eclipse o NetBeans, incluyen facilidades para formatear código Java.

Longitud de línea

La longitud de línea no debe superar los 80 caracteres por motivos de visualización e impresión.

División de líneas

Cuando una expresión ocupe más de una línea, esta se podrá romper o dividir en función de los siguientes criterios:

- Tras una coma.
- Antes de un operador.
- Se recomienda las rupturas de nivel superior a las de nivel inferior.
- Alinear la nueva línea con el inicio de la expresión al mismo nivel que la línea anterior.
- Si las reglas anteriores generan código poco comprensible, entonces estableceremos tabulaciones de 8 espacios.

Comentarios de implementación

Estos comentarios se utilizan para describir el código, y en ellos se incluye información relacionada con la implementación, tales como descripción de la función de variables locales, fases lógicas de ejecución de un método, captura de excepciones, etc. Se distinguen tres tipos de comentarios de implementación:

Comentarios de bloque

Permiten la descripción de ficheros, clases, bloques, estructuras de datos y algoritmos.

/* * Esto es un comentario

* de bloque */ Comentarios

de línea

Son comentarios cortos localizados en una sola línea y tabulados al mismo nivel que el código que describen. Si ocupa más de una línea se utilizará un comentario de bloque. Deben estar precedidos por una línea en blanco.

/* Esto es un comentario de línea */

// Esto es otro comentario de línea

Comentario a final de línea

Comentario situado al final de una sentencia de código y en la misma línea.

```
int contador = 4 + 10; // Inicialización del contador
contador++; /* Incrementamos el contador */
```

Inicialización

Toda variable local tendrá que ser inicializada en el momento de su declaración, salvo que su valor inicial dependa de algún valor que tenga que ser calculado previamente.

```
int idUnidad = 1; String[] funciones = { "Administración",
"Intervención", "Gestión" };
```

Declaración de clases

Durante el desarrollo de clases se deben seguir las siguientes reglas de formateo:

- No incluir ningún espacio entre el nombre del método y el paréntesis inicial del listado de parámetros.
- El carácter inicio de bloque "{" debe aparecer al final de la línea que contiene la sentencia de declaración.
- El carácter fin de bloque "}" se sitúa en una nueva línea tabulada al mismo nivel que su correspondiente sentencia de inicio de bloque, excepto cuando la sentencia sea nula, en tal caso se situará detrás de "{".
- Los métodos se separarán entre sí mediante una línea en blanco.

Espacios en blanco

Las líneas y espacios en blanco mejoran la legibilidad del código permitiendo identificar las secciones de código relacionadas lógicamente. Se utilizarán espacios en blanco en los siguientes casos:

- Entre una palabra clave y un paréntesis. Esto permite que se distingan las llamadas a métodos de las palabras clave. Por ejemplo:

```
while (true) {
    ... }
```

- Tras cada coma en un listado de argumentos. Por ejemplo:

```
objeto.unMetodo(a, b, c);
```

- Para separar un operador binario de sus operandos, excepto en el caso del operador (".").
  Nunca se utilizarán espacios entre los operadores unarios (p.e., "++" o "--") y sus operandos.
  Por ejemplo:

```
a += b + c;

a = (a + b) / (c + d);

contador++;
```

- Para separar las expresiones incluidas en la sentencia "for". Por ejemplo:

```
for (expresion1; expresion2; expresion3)
```

- Al realizar el moldeo o "casting" de clases. Ejemplo:

```
Unidad unidad = (Unidad) objeto;
```

Clases e interfaces

Los nombres de clases deben ser sustantivos y deben tener la primera letra en mayúsculas. Si el nombre es compuesto, cada palabra componente deberá comenzar con maýusculas. Los nombres serán simples y descriptivos. Debe evitarse el uso de acrónimos o abreviaturas, salvo en aquellos casos en los que dicha abreviatura sea más utilizada que la palabra que representa (URL, HTTP, etc.).

Las interfaces se nombran siguiendo los mismos criterios que los indicados para las clases. Como norma general toda interfaz se nombrará con el prefijo "I" para diferenciarla de la clase que la implementa (que tendrá el mismo nombre sin el prefijo "I").

class Persona class
OrganigramaDAO class
AgendaService

Métodos

Los métodos deben ser verbos escritos en minúsculas. Cuando el método esté compuesto por varias palabras cada una de ellas tendrá la primera letra en mayúsculas.

public void insertaUnidad(Unidad unidad);
public void eliminaAgenda(Agenda agenda);
public void actualizaTramite(Tramite tramite)

Variables

Las variables se escribirán siempre en minúsculas. Las variables compuestas tendrán la primera letra de cada palabra componente en mayúsculas.

Las variables nunca podrán comenzar con el carácter "_" o "$". Los nombres de variables deben ser cortos y sus significados tienen que expresar con suficiente claridad la función que desempeñan en el código. Debe evitarse el uso de nombres de variables con un sólo carácter, excepto para variables temporales.

Unidad unidad;
Agenda agenda;

Constantes

Todos los nombres de constantes tendrán que escribirse en mayúsculas. Cuando los nombres de constantes sean compuestos las palabras se separarán entre sí mediante el carácter de subrayado "_".

int LONGITUD_MAXIMA; int
LONGITUD_MINIMA;

CONVENCIONES

Clases e interfaces en Java

Los nombres de las clases deben ser sustantivos, en mayúsculas y minúsculas, con la primera letra de cada palabra interna en mayúscula. El nombre de las interfaces también debe estar en mayúscula (la primera) al igual que los nombres de las clases. Use palabras completas y debe evitar acrónimos y abreviaturas.

Generar la costumbre de escribir ciertas palabras en Inglés, es más fácil encontrar ayuda en foros o sitios en la Web. Además, mejorará el dominio con ese idioma.

Métodos en Java

Los métodos deben ser verbos, en mayúsculas y minúsculas, con la primera letra de cada palabra interna (a partir de la segunda) en mayúscula.

Variables en Java

Los nombres de las variables deben ser cortos pero significativos.

No debería comenzar con un guión bajo (‘_’) o caracteres, como por ejemplo, un signo de dólar ‘$’. Debe ser mnemotécnico, es decir, diseñado para indicar al observador casual la intención de su uso. Se deben evitar los nombres de variable de un carácter, excepto para variables temporales. Los nombres comunes para las variables temporales son: i, j, k, m y n para enteros; c, d y e para los caracteres.

Variables constantes en Java

Debería estar todo en mayúsculas con palabras separadas por guiones bajos (“_”).

Hay varias constantes utilizadas en clases predefinidas como Float, Long, String, etc.

Creando aplicaciones de consola en Java

Lo primero que se debe saber es qué tipos de programas se pueden realizar con Java. Estos son fundamentalmente tres:

- Aplicaciones de consola
- Aplicaciones de propósito general
- Applets

Las aplicaciones de consola son aquellos programas en Java que se van a ejecutar en una ventana de comandos o de Shell.

Cuando se utiliza un entorno de desarrollo (IDE) como Eclipse o Netbeans, estos incluyen su propia ventana de consola para no tener que ejecutar la que viene con el sistema operativo.

Las aplicaciones de propósito general, es decir, programas que pueden construirse para diversos objetivos o para cubrir diferentes necesidades, por ejemplo un programa hecho en Java sería Netbeans o Eclipse.

Y los applets son programas creados en Java que se ejecutan dentro de un navegador como si fuera un plugin.

En programación, se tiende a enseñar primero las aplicaciones por consola, debido a que no utilizan tantos elementos complejos que puedan confundir a quien se encuentra aprendiendo.

1. Primero que todo, se debe crear un nuevo fichero, esto se puede hacer desde el menú File > New File. 2. Seleccionar como tipo de archivo, un archivo vacío Empty Java File. 3. Al fichero se le llamará Estructura. 4. Se ha creado un archivo vacío. 5. Comenzar a escribir código en este archivo.

La declaración class, lo que hacemos es declarar una clase, el nombre de la clase debe coincidir con el nombre del archivo, en este caso Estructura.

La segunda línea es la declaración main, estamos indicando que la parte principal del programa empieza aquí, main es el punto de entrada de la mayoría de los programas, hay excepciones como por ejemplo los applets que son programas que se ejecutan como parte de una página Web, y los servlets que son programas que se ejecutan en un servidor.

Las llaves también forman parte de la estructura del programa, gracias a ellas creamos los bloques. En este ejemplo vemos que la llave de apertura de la línea uno tiene que ver con la llave de cierre de la línea cinco, forma un grupo, dentro de este grupo hay una llave de apertura y otra de cierre que forma un bloque dentro de ese grupo.
La función System.out.prinln nos va a imprimir en pantalla lo que le pasemos dentro de los paréntesis entre comillas.

Si este programa lo ejecutamos nos aparece en pantalla lo que hay entre paréntesis en la función System.out.println:

Depuración de programas utilizando el IDE XXXXXX

** Insertar la información según el IDE a utilizar

Recursos Bibliográficos: https://javadesdecero.es/fundamentos/convenciones-nomenclatura-java/