TALENTO DIGITAL

INTELIGENCIA HUMANA

UN PROYECTO DE:

Ministerio de Hacienda — Gobierno de Chile

Ministerio del Trabajo y Previsión Social — Gobierno de Chile

DESARROLLADO POR:

CORFO · sence · SOFOFA · OTIC SOFOFA Capital Humano · CPC · ACTI · BID · FCH Fundación Chile · kodea

MÓDULO 1 - PROGRAMACIÓN BÁSICA EN JAVA

1.2 EL ENTORNO JAVA PARA LA PROGRAMACIÓN

Semana 3 - Día 12

Objetivo de la jornada

- Conocer las API de Java el concepto de arreglos unidimensionales y utilizarlos para dar solución a un problema según requerimientos establecidos.
- Comprender y utilizar JavaDoc para documentar el código.

API de Java

Al instalar Java en nuestro ordenador, además del compilador y la máquina virtual de Java se instalan bastantes más elementos. Entre ellos, una cantidad muy importante de clases que ofrece Java y que están a disposición de todos los programadores listas para ser utilizadas. Estas clases junto a otros elementos forman lo que se denomina API (Application Programming Interface) de Java.

El código de estas librerías se encuentran guardadas como código cerrado o código máquina, sin poder acceder a ellas. Esto se debe a que si cada persona accediera al código fuente de los elementos esenciales de Java y los modifica, no habría compatibilidad ni homogeneidad en la forma de escribir programas Java.

Aunque no podamos acceder al código fuente, sí podemos crear objetos de los tipos definidos en la librería e invocar sus métodos. Para ello lo único que hemos de hacer es conocer las clases y la signatura de los métodos, y esto lo tenemos disponible en la documentación de Java accesible para todos los programadores a través de internet o cds de libros y revistas especializadas.

String

Dentro de un objeto de la clase String o StringBuffer, Java crea un array de caracteres de una forma similar a como lo hace el lenguaje C++. A este array se accede a través de las funciones miembro de la clase.

Los strings u objetos de la clase String se pueden crear explícitamente o implícitamente. Para crear un string implícitamente basta poner una cadena de caracteres entre comillas dobles. Por ejemplo, cuando se escribe:

System.out.println("El primer programa");
Java crea un objeto de la clase String automáticamente. Si se desea crear un string explícitamente se escribe:

String str = new String("El primer programa");

Otra forma de escribirlo sería:

String str = "El primer programa";

Para crear un string nulo se puede hacer de estas dos formas

String str=""; String
str=new String();

Un string nulo es aquél que no contiene caracteres, pero es un objeto de la clase String. Sin embargo,

String str;

está declarando un objeto str de la clase String, pero aún no se ha creado ningún objeto de esta clase.

Una vez creado un objeto de la clase String podemos obtener información relevante acerca del objeto a través de las funciones miembro. Para obtener la longitud, número de caracteres que guarda un string se llama a la función miembro length.

```
String str="El primer programa"; int
longitud=str.length();
```

Podemos conocer si un string comienza con un determinado prefijo, llamando al método startsWith, que devuelve true o false, según que el string comience o no por dicho prefijo

```
String str="El primer programa"; boolean
resultado=str.startsWith("El");
```

En este ejemplo la variable resultado tomará el valor true.

De modo similar, podemos saber si un string finaliza con un conjunto dado de caracteres, mediante la función miembro endsWith.

```
String str="El primer programa"; boolean
resultado=str.endsWith("programa");
```

Si se quiere obtener la posición de la primera ocurrencia de la letra p, se usa la función indexOf.

```
String str="El primer programa"; int
pos=str.indexOf('p');
```

Para obtener las sucesivas posiciones de la letra p, se llama a otra versión de la misma función

```
pos=str.indexOf('p', pos+1);
```

El segundo argumento le dice a la función indexOf que empiece a buscar la primera ocurrencia de la letra p a partir de la posición pos+1.

Otra versión de indexOf busca la primera ocurrencia de un substring dentro del string.

```
String str="El primer programa"; int
pos=str.indexOf("pro");
```

Vemos que una clase puede definir varias funciones miembro con el mismo nombre pero que tienen distinto número de parámetros o de distinto tipo.

Comparación de strings

La comparación de strings nos da la oportunidad de distinguir entre el operador lógico == y la función miembro equals de la clase String. En el siguiente código

```
String str1="El lenguaje Java"; String
str2=new String("El lenguaje Java");

if(str1==str2){
    System.out.println("Los mismos objetos");
}else{ System.out.println("Distintos objetos"); }
if(str1.equals(str2)){
    System.out.println("El mismo contenido"); }else{ System.out.println("Distinto contenido"); }
```

Esta porción de código devolverá que str1 y str2 son dstintos objetos pero con el mismo contenido. str1 y str2 ocupan posiciones distintas en memoria pero guradan los mismos datos.

Cambiemos la segunda sentencia y escribamos

```
String str1="El lenguaje Java";
String str2=str1;
System.out.prinln("Son el mimso objeto "+(str1==str2);
```

Los objetos str1 y str2 guardan la misma referencia al objeto de la clase String creado. La expresión (str1==str2) devolverá true.

El método equals compara un string con un objeto cualquiera que puede ser otro string, y devuelve true cuando dos strings son iguales o false si son distintos.

String str="El lenguaje Java"; boolean
resultado=str.equals("El lenguaje Java");

La variable resultado tomará el valor true.

Convertir un número a string

Para convertir un número en string se emplea la función miembro estática valueOf (más adelante explicaremos este tipo de funciones).

int valor=10; String
str=String.valueOf(valor);

La clase String proporciona versiones de valueOf para convertir los datos primitivos: int, long, float, double.

Esta función se emplea mucho cuando programamos applets, por ejemplo, cuando queremos mostrar el resultado de un cálculo en el área de trabajo de la ventana o en un control de edición.

Math

La clase Math tiene miembros dato y funciones miembro estáticas, vamos a conocer algunas de estas funciones, cómo se llaman y qué tarea realizan.

La clase Math define dos constantes muy útiles, el número pi y el número e.

public final class Math {

public static final double E = 2.7182818284590452354; public static final double PI = 3.14159265358979323846; }

El modificador final indica que los valores que guardan no se pueden cambiar, son valores constantes

Se accede a estas constantes desde la clase Math, de la siguiente forma:

```
System.out.println("Pi es " + Math.PI);
System.out.println("e es " + Math.E);
```

La clase Math define muchas funciones y versiones distintas de cada función.

Por ejemplo, para hallar el valor absoluto de un número define las siguientes funciones. Se llama a una u otra dependiendo del tipo de dato que se le pasa en su único argumento.

```
public final class Math {
    public static int abs(int a) { return (a <
        0) ? -a : a; } public static long
    abs(long a) { return (a < 0) ? -a : a; }
    public static float abs(float a) { return
    (a < 0) ? -a : a; } public static double
            abs(double a) {
      return (a < 0) ? -a : a; } //... }
```

Para hallar la raíz cuadrada de un número, se emplea la función sqrt

```
System.out.println("La raíz cuadrada de " + x + " is " + Math.sqrt(x));
```

Para expresar un número real con un número especificado de números decimales empleamos la función round. Por ejemplo, para expresar los números x e y con dos cifras decimales escribimos

```
double x = 72.3543; double y = 0.3498; System.out.println(x + " es aprox.
" + (double)Math.round(x*100)/100); System.out.println(y + " es aprox. "
+ (double)Math.round(y*100)/100);
```

Se obtiene 72.35 y 0.35 como cabría esperar. Fijarse que round devuelve un número entero int que es necesario promocionar a double para efectuar la división entre 100.

Si empleamos la función floor en vez de round obtendríamos

```
System.out.println(x + " es aprox. " + Math.floor(x*100)/100);
System.out.println(y + " es aprox. " + Math.floor(y*100)/100);
```

Se obtiene 72.35 y 0.34. La aproximación del primero es correcta ya que la tercera cifra decimal es 4 inferior a 5. La aproximación del segundo es incorrecta ya que la tercera cifra decimal es 9 mayor que 5. En la mayor parte de los cálculos se cometen errores, por lo que la diferencia entre floor y round no es significativa.

El mayor y el menor de dos números

Para hallar el mayor y el menor de dos números se emplean las funciones min y max que comparan números del mismo tipo.

```
int i = 7; int j = -9;
double x = 72.3543;
double y = 0.3498;

// para hallar el menor de dos número System.out.println("min(" + i
+ "," + j + ") es " + Math.min(i,j)); System.out.println("min(" + x + ","
+ y + ") es " + Math.min(x,y));

// Para hallar el mayor de dos números System.out.println("max(" +
i + "," + j + ") es " + Math.max(i,j)); System.out.println("max(" + x +
"," + y + ") es " + Math.max(x,y));
```

Números aleatorios

La clase Math define una función denominada random que devuelve un número pseudoaleatorio comprendido en el intervalo [0.0, 1.0). Existe otra alternativa, se pueden generar números pseudoaleatorios a partir de un objeto de la clase Random, que llame a la función miembro nextDouble.

```
System.out.println("Número aleatorio: " + Math.random());
System.out.println("Otro número aleatorio: " + Math.random());
```

Documentando el código con JavaDoc

Documentar un proyecto es algo fundamental de cara a su futuro mantenimiento. Cuando programamos una clase, debemos generar documentación lo suficientemente detallada sobre ella como para que otros programadores sean capaces de usarla sólo con su interfaz. No debe existir necesidad de leer o estudiar su implementación, lo mismo que nosotros para usar una clase del API Java no leemos ni estudiamos su código fuente.

Javadoc es una utilidad de Oracle para la generación de documentación de APIs en formato HTML a partir de código fuente Java. Javadoc es el estándar para documentar clases de Java. La mayoría de los IDEs utilizan javadoc para generar de forma automática documentación de clases. BlueJ también utiliza javadoc y permite la generación automática
de documentación, y visionarla bien de forma completa para todas las clases de un proyecto, o bien de forma particular para una de las clases de un proyecto.

Veamos en primer lugar qué se debe incluir al documentar una clase:

- Nombre de la clase, descripción general, número de versión, nombre de autores.

- Documentación de cada constructor o método (especialmente los públicos) incluyendo: nombre del constructor o método, tipo de retorno, nombres y tipos de parámetros si los hay, descripción general, descripción de parámetros (si los hay), descripción del valor que devuelve.

Las variables de instancia o de clase no se suelen documentar a nivel de javadoc.

Para generar la documentación de un proyecto automáticamente hemos de seguir unas normas a la hora de realizar los comentarios dentro del mismo. Si las hemos seguido, en BlueJ disponemos de la opción del menú Tools --> Project Documentation, que nos abre la documentación del proyecto en un navegador web. Para ver la documentación de una clase específica en BlueJ, debemos abrir la ventana de código, y en la parte superior derecha cambiar la pestaña que pone “Source code” por la opción “Documentation”.

Además de documentar las clases, todo proyecto debería tener un archivo Leeme o Readme. En BlueJ podemos acceder al readme.txt de proyecto haciendo doble click en el icono que representa una hoja en la parte superior izquierda del diagrama de clases. En el readme.txt sería adecuado incluir al menos: título del proyecto, descripción, versión, cómo arrancar el proyecto, autores e instrucciones para los usuarios. Haz doble click en el icono e introduce una descripción para tu proyecto.

Una vez cierres la ventana, la información incluida se guarda automáticamente. Para que javadoc sea capaz de generar documentación automáticamente han de seguirse estas reglas:

La documentación para javadoc ha de incluirse entre símbolos de comentario que han de empezar con una barra y doble asterisco, y terminar con un asterisco y barra simple.

/** * Esto es un comentario para javadoc */ La ubicación le define a javadoc qué representa el comentario: si está incluido justo antes de la declaración de clase se considerará un comentario de clase, y si está incluido justo antes de la signatura de un constructor o método se considerará un comentario de ese constructor o método.
Para alimentar javadoc se usan ciertas palabras reservadas (tags) precedidas por el carácter "@", dentro de los símbolos de comentario javadoc. Si no existe al menos una línea que comience con @ no se reconocerá el comentario para la documentación de la clase.

En la tabla siguiente se muestran algunas de las palabras reservadas (tags), aunque hay más de las que aquí se mencionan.

TAG	DESCRIPCIÓN	COMPRENDE
@author	Nombre del desarrollador.	Nombre autor o autores.
@deprecated	Indica que el método o clase es obsoleto (propio de versiones anteriores) y que no se recomienda su uso.	Descripción
@param	Definición de un parámetro de un método, es requerido para todos los parámetros del método.	Nombre de parámetro y descripción
@return	Informa de lo que devuelve el método, no se aplica en constructores o métodos void.	Descripción del valor de retorno
@see	Asocia con otro método o clase.	Referencia cruzada.
@version	Versión del método o clase	Versión

Las etiquetas @author y @version se usan para documentar clases e interfaces. Por tanto no son válidas en cabecera de constructores ni métodos. La etiqueta @param se usa para documentar constructores y métodos. La etiqueta @return se usa solo en métodos de tipo función.

Dentro de los comentarios se admiten etiquetas HTML, por ejemplo con @see se puede referenciar una página web como link para recomendar su visita de cara a ampliar información.

Recursos Bibliográficos: