Argentina Programa

Clase 01: Variables, tipos de datos y operadores básicos

Agenda

Familiarizarse con los conceptos básicos relacionados a la programación

- Variables, tipos de datos y operadores básicos

- Práctica Variables, tipos de datos y operadores básicos.

Definición informática de “dato”

Un dato es una representación simbólica de un atributo o una variable. Los usaremos para describir objetos a nivel programación.

Se trata de un valor o una referencia, para luego ser procesado y de esta forma construir información dentro de una solución, o en el desarrollo de un algoritmo.

Dato:
Fecha de nacimiento

Información:

- Edad
- ¿Es mayor de edad?
- ¿Puede acceder a ciertos servicios?

Tipos de Datos

- primitivos
- referenciados.

Tipos de Datos

Primitivos

Son tipos de datos con dimensión y características preestablecidas dentro del lenguaje.

Ante esto, cuando se instala alguna de estas variables, el sistema le asigna una dirección y una dimensión específicas (medida en bytes/bits).

Un ejemplo de este tipo de datos son los números enteros (integer).

Tipos de Datos

Referenciados

El principal detalle de estos datos es que no tienen dimensión específica por su condición.

Se les asigna una dirección, donde el dato comienza a guardarse, y a medida que su dimensión cambia el sistema lo actualiza.

Un ejemplo de esto serían las cadenas de caracteres (strings).

Tipos de Datos - Primitivos

Existen distintos datos primitivos en Java, y a continuación desarrollaremos los siguientes:

- enteros,
- flotantes,
- char.

Tipos de Datos - Primitivos

Enteros

Como su nombre lo indica, son espacios asignados para almacenar números enteros.

El espacio ocupado por cada uno depende pura y exclusivamente del tipo específico a utilizar:

- Byte (1 byte, 8 bits): almacena números que van desde -128 hasta 127
- Short (2 bytes): almacena números que van desde -32768 hasta 32767
- Integer (4 bytes): puede almacenar números muy grandes.
- Long (8 bytes): puede almacenar números extremadamente grandes.

Tipos de Datos - Primitivos

Enteros

Como su nombre lo indica, son espacios asignados para almacenar números enteros.

El espacio ocupado por cada uno depende pura y exclusivamente del tipo específico a utilizar:

- Byte (1 byte, 8 bits): almacena números que van desde -128 hasta 127
- Short (2 bytes): almacena números que van desde -32768 hasta 32767
- Integer (4 bytes): puede almacenar números muy grandes.
- Long (8 bytes): puede almacenar números extremadamente grandes.

Tipos de Datos - Primitivos

Flotantes

Es una variable también numérica, pero que admite parte decimal, así como también números negativos.

Su mayor complejidad hará que ocupe más espacio que una variable del tipo entera, existiendo también distintos tipos de float en Java, según el grado de precisión necesaria:

- Float (4 bytes): almacena números grandes, con parte decimal.
- Double (8 bytes): almacena números muy grandes, con parte decimal.

Tipos de Datos - Primitivos

Char

En Java, los caracteres no están establecidos a partir del código ASCII, sino que emplean **Unicode**, almacenando en estas variables cualquier carácter del código, así como las distintas secuencias de escape(\n, \r, \t, etc.).

Una variable tipo char ocupa 2 bytes.

Tipos de Datos - Referenciados

Una base para las colecciones

Como ya mencionamos, un string es una variable referenciada.

Almacenar variables a partir de posiciones de memoria, donde lo alojado comprenderá desde una dirección inicial, hasta una dirección final y esta longitud es variable, es una base primordial para el concepto de colecciones (**collections**).

Tipos de Datos - Referenciados

Colecciones

Pensemos en el significado coloquial de la palabra: una colección es un conjunto de elementos que cumplen con determinadas características o parámetros; los elementos almacenados tienen algo en común.

Podemos hablar de una colección de discos, pinturas, libros, o en este caso datos u objetos.

Tipos de Datos - Referenciados

Colecciones

Las colecciones están desarrolladas a partir de interfaces, y estas implican abstracción, es decir, nos proveen de estructuras confeccionadas para un uso genérico, y quedará en manos del/la desarrollador/a la implementación de las mismas.

Tipos de Datos - Referenciados

Colecciones

¿QUÉ COLECCIÓN USAR?

Para realizar un correcta selección, debemos preguntarnos lo siguiente:

- ¿Necesito un orden estático? ¿Puedo reordenar los elementos?
- ¿Hay duplicados?
- ¿Con qué frecuencia agregaremos elementos?

Collections - Arrays

Es una estructura de almacenamiento de datos completamente básica.

Permite manipular información de manera muy veloz, ya que su forma de organización, replica el formato de un banco de memoria.

Podemos instanciar arreglos de distintos tipos de variables, incluso arreglos de objetos customizados.

Collections - List

Se caracteriza por ser secuencia de elementos dispuesto en un cierto orden, en la que cada elemento tiene como mucho un predecesor y un sucesor.

Podríamos decir que en una lista, por lo general, el orden es dato, es decir, el orden es información importante que la lista también nos está almacenando.

Collections - List

TIPOS DE LISTAS

- **ArrayList:** similar a los arreglos antes descritos, con la salvedad de que estos crecen de manera dinámica. Por su organización, se vuelve costoso la eliminación de elementos.
- **LinkedList:** al ser enlazada, los elementos guardan un vínculo con sus “vecinos”, haciendo muy sencilla la eliminación, pero complejo el recorrido de la misma.

Collections - Set

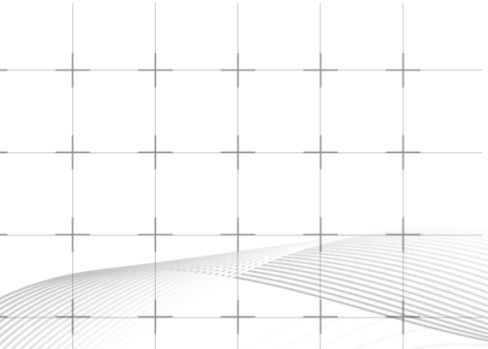
Esta colección está pensada para lo que matemáticamente se conoce como conjunto, lo cual agrega un detalle clave: no puede haber duplicados.

A su vez, en este tipo de colección el orden deja de ser un detalle significativo, por lo que la interfaz carece de elementos relacionados al mismo.

Collections - Set

TIPOS DE CONJUNTOS

- HashSet
- TreeSet
- Map



Collections - Set

HashSet

Es la más comúnmente usada, e implica el almacenamiento de elementos a los cuales se les asocia un “hash”.

Este último es básicamente un número entero, que sirve de referencia para almacenar elementos en una suerte de “tabla”.

Collections - Set

TreeSet

Esta técnica construye un árbol (tree) con los objetos que se van agregando al conjunto, teniendo un elemento “raíz”, y nodos que se van desprendiendo como ramificaciones (de la misma manera que en un árbol genealógico).

Es fácil de recorrer, pero costoso de mantener organizado.

Collections - Set

Map

Conjunto de objetos, con el detalle de que cada uno de estos valores tiene un objeto extra asociado (manteniendo el paralelo con el diccionario, podemos decir que en este, cada palabra, conlleva una definición asociada).

A los primeros se los llama “claves” o “keys”, ya que nos permiten acceder a los segundos.

Collections

Manteniendo conceptos recientemente enunciados, el conjunto de claves corresponde técnicamente a un Set, no puede haber duplicados.

De todas formas, un Map no es una Collection ya que esa interfaz le queda demasiado chica.

Podríamos decir que Collection es unidimensional, mientras que un Map es bidimensional.

Genéricos

Dotan al lenguaje de mayor versatilidad, aumentando la reutilización de código, al mismo tiempo qué reducen los distintos tipos procesos de casteos.

```
public class Box {
	private T t;
	public T get() {
		return t;
	}
	public void set(T t) {
				this.t = t;
	}
}
```

Genéricos

Genéricos en Java Según las convenciones los nombres de los parámetros de tipo usados comúnmente son los siguientes:

- E: elemento de una colección.
- K: clave.
- N: número.
- T: tipo.
- V: valor.
- S, U, V etc: para segundos, terceros y cuartos tipos.

Operadores básicos java

- **Operadores aritméticos:**+ (suma), - (resta), * (multiplicación), / (división) y % (módulo)

- **Operadores de asignación:** = (igual)

- **Operadores de comparación:** == (igual que), != (distinto que), > (mayor que), < (menor que), >= (mayor o igual que) y <= (menor o igual que).
- Operadores lógicos: AND, OR y NOT.
  && (y lógico), || (o lógico) y ! (negación lógica).

- **Operadores de incremento y decremento:** ++ (incremento), -- (decremento)

- **Operadores de asignación compuesta**

Operadores básicos java

Operator Type	Category	Precedence	Associativity
Unary	postfix	a++, a--	Right to left
	prefix	++a, --a, +a, -a, ~, !	Right to left
Arithmetic	Multiplication	*, /, %	Left to Right
	Addition	+, -	Left to Right
Shift	Shift	<<, >>, >>>	Left to Right
Relational	Comparison	<, >, <=, >=, instanceOf	Left to Right
	equality	==, !=	Left to Right
Bitwise	Bitwise AND	&	Left to Right
	Bitwise exclusive OR	^	Left to Right
	Bitwise inclusive OR	|	Left to Right
Logical	Logical AND	&&	Left to Right
	Logical OR	||	Left to Right
Ternary	Ternary	? :	Right to Left
Assignment	assignment	=, +=, -=, *=, /=, %=, &=, ^=, |=, <<=, >>=, >>>=	Right to Left

Imagen desde https://javagoal.com/operators-in-java/

Scanner

Como en otros lenguajes, el verbo “scan” se asocia a leer valores desde el teclado.

Esto nos permitirá el ingreso de datos, asociados a las variables que estudiamos con anterioridad.

También es importante saber que:

- Scanner es una clase dentro del paquete java.util,
- Permite trabajar con archivos.
- Para emplearlo debemos crear un objeto, y luego cerrar el scanner.

Scanner

Uso

Creamos el componente: Scanner sc = new Scanner(System.in);

Empleamos para tomar un Double: Double numero1 = sc.nextDouble();

Empleamos para tomar un Char: char operacion = sc.next().charAt(0);

Cerramos el scanner: sc.close();

Genéricos

Dotan al lenguaje de mayor versatilidad, aumentando la reutilización de código, al mismo tiempo qué reducen los distintos tipos procesos de casteos.

```
public class Box {
	private T t;
	public T get() {
		return t;
	}
	public void set(T t) {
				this.t = t;
	}
}
```

CONSULTAS?

Muchas Gracias!