Comandos Java

|  |  |  |
| --- | --- | --- |
| Código | Uso | Ejemplo |
| JFrame | Crear ventana | **JFrame** miVentana = new JFrame(); |
| JButton | Colocar botones a una ventana. | import javax.swing.\*;  **JButton** boton1 = new JButton(“Opción 1”); |
| setSize() | Determinar tamaño de la ventana. | jFrame.**setSize**(400, 300); |
| setTitle() | Definir título de la ventana. | jFrame.**setTitle**("Mi primera ventana"); |
| setLocation() | Definir lugar en el que se ubica la ventana en pantalla. | jFrame.**setLocation**(150, 250); |
| setBackground() | Definir un color de fondo diferente.  Nota 1: es necesario acceder al panel de contenido (content pane), para ello llamamos al método getContentPane(). | Container contenidoPanel = jFrame.getContentPane();  contenidoPanel.**setBackground**(Color.MAGENTA); |
| import | Referirse a clases definidas en el paquete designado sin usar el nombre de clase completamente calificado. | import javax.swing.\*; |
| showMessageDialog() | Método de la clase JOptionPane. Sirve para mostrar por pantalla un mensaje determinado.  Nota 1: necesita de un JFrame.  Nota 2: JOptionPane sólo recibe String como entrada, de modo que para ingresar un valor numérico debemos realizar la conversión nosotros mismos con las funciones “parse”. | JFrame jFrame = new JFrame(); jFrame.setSize(400, 300); jFrame.setVisible(true);  JOptionPane.**showMessageDialog**(jFrame, "Hola Caro!"); |
| showInputDialog() | Método de la clase JOptionPane. Sirve para ingresar texto por pantalla. | JFrame jFrame = new JFrame(); jFrame.setSize(400, 300); jFrame.setVisible(true);  String texto;  texto = JOptionPane.**showInputDialog**(jFrame, "Introduzca su nombre: "); |
| parseInt()  parseLong()  parseFloat()  parseDouble() | Funciones utilizadas para convertir cadenas en tipos numéricos como int, long, float y double. | String texto2;  texto2 = JOptionPane.showInputDialog(jFrame, "Edad: ");  int edad = **Integer.parseInt**(texto2); |
| FlowLayout | Manejador de presentación (objeto que controla la colocación de los objetos GUI). |  |
| System.out.print(); | Imprime resultados por la consola  Nota 1: poner System con S mayúscula  Nota 2: la consola no es la pantalla!!!  Nota 3: println (print line) es para que haya un enter entre distintos print | System.out.print("Hola soy Carolina Bottino"); |
| substring(i, j) | Para extraer subcadenas de una cadena |  |
| length() | Para conocer la longitud de la variable tipo string llamada “texto” | System.out.println(texto.length()); |
| indexOf() | Para conocer la posición en la cual empieza una subcadena dentro de una cadena. |  |
| charAt | Permite acceder a los caracteres individuales de una cadena. |  |
| toUpperCase() | Función de la clase útil que convierte las letras de una cadena en mayúsculas. | String nombre = “carolina”, nombreEnMayusculas;  nombreEnMayusculas = nombre.**toUpperCase()**; |
| replaceAll() | Método que sirve para remplazar todas las subcadenas de una cadena por una cadena de remplazo dada.  Nota 1: si lo ponemos así como está, en aquellos casos donde yo quiera remplazar palabras como “ej” por “ejemplo”, si encuentra una cadena como “ejercicio” la va a remplazar por “ejemploercicio” entonces para simbolizar que es sólo la subcadena “ej” lo que hacemos es delimitarla con **\b**. | String texto = “su texto aquí”;  texto.**replaceAll**(“[aeiou]”, “@”);  - remplaza todas las vocales por el símbolo @ -  texto.**replaceAll**(“OOP”, “object-oriented programming”);  - remplaza todas las apariciones de OOP con object-oriented programming –  --------------  texto.**replaceAll**(“\**\b**ej\**\b**”, “ejercicio”);  Van 2 barras invertidas porque la primera simboliza que “\b” es el delimitador de la palabra y no una parte de la subcadena. |
| Operaciones con cadenas | Operaciones con cadenas |  |
| Manipulación de cadenas  StringBuffer  StringBuilder | Operaciones que permiten sustituir un carácter, añadir una cadena a otra, borrar una porción de una cadena, etc.; SIN CREAR UNA CADENA AUXILIAR.  Nota 1: StringBuffer y StringBuilder son exactamente iguales, la única diferencia es que el segundo pertenece a versiones más nuevas de Java SDK. | **StringBuffer** palabra = new StringBuffer(“Java”);  palabra.setCharAt(0, “D”);  palabra.setCharAt(1, “i”);  Convierte “Java” en “Diva”.  - ver ejemplos en pág. 516 del libro. - |
| toString() | Convierte un formato interno en una representación de cadena | System.out.println(hoy.toString()); |
| Date | Clase del paquete java.util que contiene todo lo concerniente a fechas.  Nota 1: necesita ser declarado arriba como **import java.util.Date;** |  |
| SimpleDateFormat | Clase del paquete java.text que convierte fechas a formatos más simples.  Nota 1: importar paquete **import java.text.\*;**  Nota 2: ver más detalle en pág. 60 |  |
| equals() | Devuelve true si dos objetos String tienen la misma secuencia exacta de caracteres. | String str1 = “hola”, str2 = “hola”, str3 = “Hola”;  str1.equals(str2) devuelve TRUE.  Str1.equals(str3) devuelve FALSE. |
| equalsIgnoreCase() | Hace lo mismo que equals solo que la comparación es no sensible a las mayúsculas. | String str1 = “Hola”, str2 = “hola”;  str1.equals(str2) devuelve TRUE. |
| Formatter | Clase de formateo de propósito general. | Leer libro pág. 326 a 330. |
| This | Apunta al objeto en cuestión. Si yo llamo un método con un objeto f1 y quiero invocar otro método de f1 desde allí, lo que hago es usar this.métodoF1() | Objeto de tipo “Fraccion” que tiene numerador y denominador.  Si en el main tengo la sentencia:  f1 = f1.suma(f2);  ----------------------------------------------------------  public Fraccion suma(Fraccion f2){  int a, b, c, d;  a = **this**.getNumerador(); //num f1  b = **this**.getDenominador(); //den f1  c = f2.getNumerador();  d = f2.getDenominador();  } |
| Package | Para crear paquetes con clases que luego usaremos en otros programas. Esto simplifica la reutilización de código ya que no tenemos que copiar y pegar archivos en las carpetas de los nuevos programas, simplemente importamos como con las clases y paquetes predefinidos. | **package** cap2ejemplo1;  **public** class Cap2Ejemplo1 {  …….  }  ----------------------------------------------------------  Nota 1: es importante que el nombre del package sea todo en minúscula, es una convención.  Nota 2: la clase debe ser si o si public. |
| Comentario Javadoc | Se utilizan como encabezado de las clases generalmente.  @author : etiqueta para señalar autores del documento.  @param: etiqueta para describir un parámetro.  @return: etiqueta para describir los retornos de los métodos. | /\*\*  \* @author Caro  \*/  ---------------------------  @param <nombre parámetro> <descripción>  ---------------------------  @return <descripción> |
| Scanner | Para poder ingresar datos por teclado a través de la consola. |  |
| Try-catch | Bloque para manejar excepciones.  Nota 1: todo lo que yo ponga en try se va a ejecutar en el curso normal del programa, sólo va a ejecutarse catch cuando try genere un error. | **Try**  { <sentencia/s que generen error> }  **catch**(InputMismatchException e)  { <lo que quiero que se haga cuando pase lo de arriba } |
| (int) X  (char) 88 | Permite visualizar el valor ASCII de X.  Permite visualizar la letra representada por el 88 en la tabla de valores ASCII | System.out.println(“El código ASCII de la letra X es: “ + (int) ‘X’ );  System.out.println(“La letra que corresponde al ASCII número 88 es: “ + (char) 88 ); |
| HERENCIA | En el ejemplo, la clase “Gato” es una subclase de “Mascota”  - ver libro capítulo 13 - | **Class** Gato **extends** Mascota{  //atributos  //métodos  }  ----------  Formas de declarar válidas:  - Mascota m1 = new Mascota();  - Mascota m2 = new Gato(); |
|  |  |  |

Sentencias de Java

Hoy vamos a ver que existen dos tipos de sentencias de decisiones en Java. Veamos cuales son:

* La sentencia **if**.
* La sentencia **switch**.

## La sentencia if

Una sentencia if consiste en una expresión booleana seguida de una o más sentencias. Veamos la sintaxis:

if(expresion\_Booleana)

{

//Sentencias se ejecutan si la expresión booleana es verdadera

}

Si la expresión booleana es verdadera, entonces el bloque de código dentro del if se ejecuta. Si no se ejecutará el primer grupo de código después del final de la instrucción if.

Veamos un ejemplo:

public class Curso {

public static void main(String args[]){

int x = 15;

if( x < 30 ){

System.out.print(“Esto es una sentencia if");

}

}

}

Si ejecutamos el código anterior obtendremos lo siguiente:

Esto es una sentencia if

## La sentencia if..else

Una sentencia **if** puede ser seguido por una sentencia **else**, que se ejecuta cuando la expresión booleana es falsa.

Veamos la sintaxis:

if(expresion\_Booleana)

    / / Se ejecuta cuando la expresión booleana es verdadera

} else {

    / / Se ejecuta cuando la expresión booleana es falsa

}

Veamos un ejemplo:

public class Curso {

public static void main(String args[]){

int x = 40;

if( x < 10 ){

System.out.print("Esto es una sentencia if");

}else{

System.out.print("Esto es una sentencia else");

}

}

}

Si ejecutamos el código anterior obtendremos lo siguiente:

Esto es una sentencia else

## La sentencia if...else if…else

Una sentencia if puede ser seguido por un opcional else if ... else, que es muy útil para comprobar varias condiciones.

Veamos la sintaxis:

if(expresion\_Booleana\_1)

    / / Se ejecuta cuando la expresión booleana 1 es verdadera

} else if (expresion\_Booleana\_2) {

    / / Se ejecuta cuando la expresión booleana 2 es verdadera

} else if (expresion\_Booleana\_3) {

    / / Se ejecuta cuando la expresión booleana 3 es cierto

} else {

    / / Se ejecuta cuando ninguna condición anterior es verdadera.

}

Veamos un ejemplo:

public class Curso {

public static void main(String args[]){

int x = 45;

if( x == 15 ){

System.out.print(“El valor de X es15");

}else if( x == 30 ){

System.out.print("El valor de X es 30");

}else if( x == 45 ){

System.out.print("El valor de X es 45");

}else{

System.out.print(“X no cumple ninguna de las condiciones anteriores");

}

}

}

Si ejecutamos el código anterior obtendremos lo siguiente:

El valor de X es 45

## La sentencia anidada if...else

También es posible usar una sentencia **if** o **if…else** dentro de otro **if** o **if..else**. Veamos la sintaxis:

if(expresion\_Booleana\_1)

    / / Se ejecuta cuando la expresión booleana 1 es verdadera

    if (expresion\_Booleana\_2) {

       / / Se ejecuta cuando la expresión booleana 2 es verdadera

    }

}

Veamos un ejemplo:

public class Curso {

public static void main(String args[]){

int x = 40;

int y = 5;

if( x == 40 ){

if( y == 5 ){

System.out.print("X = 30 y Y = 5");

}

}

}

}

Si ejecutamos el código anterior obtendremos lo siguiente:

X = 40 y Y = 5

## La sentencia switch

Una sentencia **switch** permite a una variable ser probada por una lista de condiciones. Cada condición se llama **case**.Veamos la sintaxis:

switch (expresion) {

     case valor1:

        / / Declaraciones

        break; / / opcional

     case valor2:

        / / Declaraciones

        break; / / opcional

     / / Usted puede tener cualquier número de sentencias case.

     default: / / Opcional

        / / Declaraciones que cumplirá si la variable no entra en ningún caso.

}

Las siguientes reglas se aplican a una sentencia switch :

* La variable que se utiliza en una sentencia switch sólo puede ser un byte, short , int, o char.
* Puedes tener cualquier número de sentencias case dentro de un switch. Cada caso es seguido del valor a ser comparado.
* El valor de un caso debe ser el mismo tipo de datos que la variable en el switch.
* Cuando la variable del switch es igual a un caso, las instrucciones que siguen a ese caso se ejecutará hasta que se alcanza una sentencia break.
* Cuando se llega a una sentencia break, el caso termina, y el flujo de control pasa a la siguiente línea después de la sentencia switch.
* No todos los casos tiene que contener un break.
* Una sentencia switch puede tener un caso por defecto (opcional), que debe aparecer al final del switch. El caso por defecto se puede utilizar para realizar una tarea cuando ninguno de los casos es cierto.

Veamos un ejemplo:

public class Curso {

public static void main(String args[]){

char departamento = 'B';

switch(departamento)

{

case 'A' :

System.out.println("Desarrollo");

break;

case 'B' :

System.out.println(“Recursos Humanos");

break;

case 'C' :

System.out.println("Finanzas");

break;

case 'D' :

System.out.println("Mercadeo");

default :

System.out.println(“Departamento invalido");

}

System.out.println(“Código para el departamento es " + departamento);

}

}

Si ejecutamos el código anterior obtendremos lo siguiente:

Recursos Humanos

Código para el departamento es B