Desarrollo Avanzado de Aplicaciones I

Introducción a Java IV

idat

i d a t

inicio desarrollo aplicación término

idat

Inicio

Logro de aprendizaje - Introducción

idat

Logro de Aprendizaje

“Al finalizar la sesión, el participante podrá desarrollar estructuras repetitivas, haciendo uso de acumuladores y contadores.”

idat

Introducción

- Revisión Rápida de Temas de Sesión Anterior
- Revisión de Ejercicios de Sesión Anterior
- Inquietudes y/o Preguntas
- Agenda de Sesión
  - Acumuladores y contadores
  - Estructuras repetitivas:
    - for
    - while
    - do ... while
  - Break y continue

Desarrollo

Desarrollo del Contenido de la Sesión

idat

Acumuladores y Contadores

idat

- ✓ Un acumulador es una variable que se utiliza para acumular o totalizar cantidades.
  - totalSuma = totalSuma + 100;
  - totalSuma += 100;
  - totalSueldo = sueldoBase + bonificación + comisiones – descuentos;
- ✓ Un contador es una variable que se utiliza para contar el número de ocurrencias de un suceso o el número de veces que se cumple una determinada condición.
  - cont = cont + 1 ;
  - cont += 1 ;
  - cont++ ;

Estructuras Repetitivas

idat

- ✓ Se denominan estructuras repetitivas a aquellas estructuras que permiten repetir instrucciones.
- ✓ A las estructuras repetitivas se conocen también como estructuras iterativas o bucles.
- ✓ A las instrucciones a repetir se conocen como el cuerpo del bucle (bloque de código) y al hecho de repetir la secuencia de instrucciones se denomina iteración.
- ✓ Tenemos tres tipos de estructuras repetitivas:
  - ➢ for
  - ➢ while
  - ➢ do ... while

Sentencia for

- La estructura de repetición “for” permite ejecutar un grupo de sentencias de forma iterativa y controlada por un contador.
- En su declaración se conoce el número de veces que se repetirá el grupo de sentencias.
- Estructura de la sentencia:
  - for (inicialización; terminación; actualización) {
  - sentencias
  - }
- Donde:
  - inicialización: tipo_de_dato contador = valor_inicial
  - terminación: contador operador_de_comparación valor_final
  - actualización: incremento o decremento

Sentencia for (cont.)

- ✓ El contador que se usa se puede declarar en la misma sentencia y se comportaría como una variable local. Generalmente es del tipo de dato "int", pero se puede usar otro tipo de dato numérico.
- ✓ Los operadores de comparación que se deben usar son: mayor que (>), menor que (<), mayor o igual que (>=) y menor o igual que (<=).
- ✓ Cuando el grupo de sentencias a repetir es solo una, se pueden obviar las llaves "{}" del bloque de código.
- ✓ Ejemplo: Imprimir los números del 1 al 9.
  - ➢ for (int i = 1 ; i < 10 ; i++)
  - ➢ System.out.println(i) ;

Sentencia for (cont.)

idat

- ✓ Se puede usar bucles “for” anidados.
- ✓ Ejemplo: Imprimir los números del 00 al 99.
  - ➢ for (int i = 0 ; i < 10 ; i++)
  - ➢ for (int j = 0 ; j < 10 ; j++)
  - ➢ System.out.println(Integer.toString(i) + Integer.toString(j));
- ✓ Tanto la sentencia de inicialización como la de actualización pueden tener varias sentencias separadas por comas.
- ✓ Ejemplo: Imprimir las combinaciones de números que suman 9.
  - ➢ for (int i = 0 , j = 9 ; i < 10 ; i++ , j--)
  - ➢ System.out.println(Integer.toString(i) + “y” + Integer.toString(j));

Sentencia while

idat

- ✓ La estructura de repetición “while” permite ejecutar un grupo de sentencias de forma iterativa e indeterminada y está controlada por una condición booleana.
- ✓ Primero se evalúa la condición booleana y de ser el resultado de la expresión verdadero se ejecuta el grupo de sentencias del bloque de código.
- ✓ Estructura de la sentencia:
  - ➢ while (condición) {
  - ➢ sentencias
  - ➢ }

Sentencia while (cont.)

- ✓ Las variables que se usen en la condición deben estar declaradas e inicializadas previamente.
- ✓ Cuando el grupo de sentencias a repetir es solo una, se pueden obviar las llaves "{}" del bloque de código.
- ✓ Ejemplo: Imprimir los números del 1 al 9.

```
int i = 1 ;
while ( i < 10 ) {
    System.out.println( i ) ;
    i++ ;
}
```

Sentencia do ... while

idat

- ✓ La estructura de repetición “do ... while” permite ejecutar un grupo de sentencias de forma iterativa y controlada por una condición booleana.
- ✓ A diferencia de la sentencia “while” primero ejecuta las sentencias del bloque de código y luego evalúa la condición booleana.
- ✓ La sentencia “do ... while” debe terminar en “;”
- ✓ Estructura de la sentencia:
  - ➢ do {
  - ➢ sentencias
  - ➢ } while (condición) ;

Sentencia do ... while

idat

- ✓ Las variables que se usen en la condición deben estar por lo menos declaradas previamente.
- ✓ Cuando el grupo de sentencias a repetir es solo una, se pueden obviar las llaves "{}" del bloque de código.
- ✓ Ejemplo: Imprimir los números del 1 al 9.
  - ➢ int i = 1 ;
  - ➢ do {
  - ➢ System.out.println(i) ;
  - ➢ i++ ;
  - ➢ } while (i < 10) ;

Break

- ✓ La palabra reservada “break” se utiliza para detener completamente un bucle y que se ejecute la siguiente línea de código fuera de la sentencia repetitiva.
- ✓ Ejemplo: Imprimir los números del 1 al 9. Interrumpir la impresión si “i” es igual a “j”.

```
int i = 1, j = 5 ;
while ( i < 10 ) {
    if ( i == j ) {
        break ; }
    System.out.println( i ) ;
    i++ ;
}
```

Continue

- ✓ La palabra reservada “continue” se utiliza para detener únicamente la iteración actual y saltar a la siguiente iteración. Se salta a la declaración de la condición o ciclo.
- ✓ Ejemplo: Imprimir los números del 1 al 9 con excepción del 5.

```
for ( int i = 1 ; i < 10 ; i++ ) {
    if ( i == 5 ) {
        continue ; }
    System.out.println( i ) ;
}
```

Aplicación

Revisar ejemplos y realizar ejercicios prácticos

idat

Término

Indicaciones generales y/o Resumen de Sesión

idat

Resumen de Sesión

idat

- Acumuladores
- Contadores
- Estructuras repetitivas
- Sentencia for
- Sentencia while
- Sentencia do ... while
- Ejemplos y ejercicios

GRACIAS