PROGRA❤AMANDO

holamundo.co

Entrada y salida de datos

Existen diferentes formas de solicitar datos a un usuario, vamos a revisar: La petición de datos por la consola y por medio de un cuadro de dialogo que emplea la interfaz gráfica de usuario (GUI:Graphic User Interface) de java.

La entrada de datos se realizará por teclado, es importante tener presente que: TODOS LOS DATOS QUE SE LEEN POR TECLADO SON DE TIPO CADENA (STRING), ya que con el teclado es posible escribir cualquier combinación de datos alfanúmericos.

Scanner

Es un escaner de texto simple que puede leer cadenas y convertir a diferentes tipos de datos primitivos.

```
import java.util.Scanner;    //Se debe importar el paquete
```

Se usa la función *nextLine*, para pedir una cadena, también hay otras variaciones como: *nextInt*, *nextLong*,... para otros tipos.

```
Scanner in = new Scanner(System.in);
System.out.print("Ingrese un dato : ");
String dato = in.nextLine();
System.out.println("El dato ingresado es : " + dato);
```

GUI

Abre un cuadro de dialogo, con un campo de texto para que el usuario ingrese un dato.

```
import javax.swing.*;        //Se debe importar el paquete
```

Se usa la función showInputDialog, para mostrar una ventana con un campo de texto al usuario, para que ingrese un dato.

```
String mensaje = "Ingrese un dato", str;
str = JOptionPane.showInputDialog(null,mensaje);
System.out.println("El dato ingresado es : " + str);
```

Excepción

Una excepción o Exception en java, es un error que se presenta durante la ejecución del programa, puede ser ocasionada por errores del programador, errores del sistema o problemas relacionados con el funcionamiento del programa.

Las excepciones son lanzadas por el programa, como respuesta a un procedimiento que puede ser grave para la continuación del programa, el programador debe estar pendiente de la aparición de una excepción durante el tiempo de desarrollo y hacer lo posible por corregirla, ya que si se produce una excepción el programa se detiene inmediatamente.

Las excepciones no siempre se pueden evitar desde código porque son ocasionadas por un ente externo al programa.

Excepciones frecuentes

```
5/0                                ArithmethicException

String str=null;
str.toUpperCase();                 NullPointerException

int i[] = {2};
i[8] = 5;                          ArrayIndexOutOfBoundsException
```

BufferReader

Es un wrapper, que optimiza la lectura de texto desde una fuente como: Un archivo, la consola, etc.

Puede causar una excepción, por tanto es necesario especificarlo en la función que realiza la lectura de datos.

```
import java.io.*;                    //Se debe importar el paquete
```

```
public static void leer() throws Exception {
  BufferedReader br;
  br = new BufferedReader(new InputStreamReader(System.in));
  System.out.print("Enter String");
  String s = br.readLine();
  System.out.println("El dato ingresado es : " + s);
}
```

Try – catch – finally

Permite atrapar una excepción y determinar que hacer con esta en el mismo momento en que se produce, sirve para evitar que una excepción generada por un ente externo termine elprograma abruptamente. Es una protección contra algunos errores que se presentan frecuentemente.

```
try {
  // Lo que puede producir una excepción
} catch(Exception e) {
  // Como se va a manejar la excepción
} finally {
  // Esto siempre se ejecuta, como control de posibles daños
}
```

Múltiples excepciones

Un bloque de código puede generar diferentes tipos de excepciones, por lo general la respuesta a cada excepción es diferente, por tanto se puede especificar que hacer con cada excepción generada. El *finally* es opcional.

```
String dato;
int numero;
  try{
        dato = JOptionPane.showInputDialog(null,"...");
        numero = Integer.parseInt(dato.length()+"");
   }catch(NumberFormatException e){
        System.out.println("Error");
   }catch(NullPointerException e){
        System.out.println("No se leyó el dato correctamente");
   }
```

Parse

Intenta convertir un dato de tipo cadena a su equivalente en un dato primitivo, si es posible retorna el dato convertido, sino se produce una excepción.

```
String dato = "123";
int numero;
  try{
        numero = Integer.parseInt(dato);
   }catch(NumberFormatException e){
        System.out.println("Error");
   }finally{
        System.out.println("Bien o mal se ejecuta");
   }
```

Errores frecuentes

Solicitar un número por teclado y usar la entrada directamente, recuerde que el usuario aunque se le este pidiendo un número se puede equivocar al digitar la entrada o simplemente no lee las indicaciones y termina ingresando un dato incorrecto, por eso se debe tener precaución y asegurarse que el dato ingresado sea compatible con un formato númerico.

No presentar un mensaje al usuario para orientarlo sobre que se espera que ingrese para que el programa pueda operar.

Usar mensajes muy largos, si esta en la consola tener que hacer scroll va en contra de la experiencia de usuario, o si es usando la interfaz gráfica, mucho texto puede ser omitido por el usuario.

Ejercicios

Realice una función que permita leer un dato por teclado de manera discreta, como pasa cuando se digita una contraseña.

Realice una función que lea desde la consola diferentes tipos de datos primitivos, enteros, flotantes, boolean, char, etc.

Realice un menú “infinito” que se le imprima al usuario mostrando una serie de opciones para escoger, si el usuario ingresa -1 el programa termina, sino se realiza alguna de las opciones disponibles, seleccionada por el usuario (Realice control de errores).

Realice un programa que pida un dato al usuario, si ingresa :), imprime “hola estás feliz” o :(, imprime, “¿por qué estas triste?.