ESCUELA DE
INGENIERÍA EN CIENCIAS Y SISTEMAS
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

ECYS | DIDT
DESARROLLO DE TRANSFERENCIA TECNOLÓGICA

Día, Fecha:	31/08/2023
Hora de inicio:	09:00

Introducción a la Programación y Computación 1 [Sección B]

Diego Fernando Cortez Lopez

Agenda

- Clase 9
- Dudas
- Ejemplo Práctico
- Corto 2

Manejo de Archivos

Clase 6

Archivos

Un archivo es un objeto en una computadora que puede almacenar información, configuraciones o comandos, el cual puede ser manipulado como una entidad por el sistema operativo o cualquier programa o aplicación.

Un archivo debe tener un nombre único dentro de la carpeta que lo contiene y cada nombre contiene un sufijo (extensión) que puede identificar el tipo de archivo.

Flujo de datos

El flujo de datos se refiere a cómo se transfieren los datos entre diferentes componentes del sistema que están involucrados en el proceso de lectura y escritura de archivos.

Las entradas y salidas de datos en java se manejan mediante streams (flujo de datos). Un stream es una conexión entre el programa y la fuente (lectura) o el destino (escritura) de los datos. La información se traslada en serie a través de esta conexión.

Flujo de datos

En Java existen 4 jerarquías de clases relacionadas con los flujos de entrada y salida de datos.

- **Flujos de bytes:** las clases derivadas de *InputStream* (para lectura) y de *OutputStream* (para escritura), las cuales manejan los flujos de datos como stream de bytes.
- **Flujos de caracteres:** las clases derivadas de *Reader* (para lectura) y *Writer* (para escritura), las cuales manejan stream de caracteres.

Clase File

La clase File permite manejar archivos o carpetas.

Cuando se crea una instancia de la clase File no se crea ningún archivo, solo se crea una referencia a un objeto de este tipo. Algunos de los métodos mas útiles que posee la clase File son:

- exists()
- createNewFile()
- mkdir()
- delete()
- renameTo()
- list()

FileOutputStream

Permite crear y escribir un flujo de bytes en un archivo de texto plano.

FileOutputStream(String nombre) o
FileOutputStream(File archivo)

```java
import java.io.FileOutputStream;
import java.io.IOException;

public class ClaseFileOutputStream {

    public static void main (String [] args){
        FileOutputStream fos = null;
        byte[] buffer = new byte[81];
        int nBytes;
        try {
            System.out.println("Escribir el texto a guardar en el archivo:");
            nBytes = System.in.read(buffer);
            fos = new FileOutputStream("fos.txt");
            fos.write(buffer,0,nBytes);
        } catch (IOException ioe){
            System.out.println("Error: " + ioe.toString());
        } finally {
            try {
                if (fos != null) fos.close();
            } catch (IOException ioe) {
                System.out.println("Error al cerrar el archivo.");
            }
        }
    }

}
```

FileInputStream

Permite leer flujos de bytes desde un archivo de texto plano

FileInputStream(String nombre) o
FileInputStream(File archivo)

```java
import java.io.FileInputStream;
import java.io.IOException;

public class ClaseFileInputStream {
    public static void main (String [] args){
        FileInputStream fis = null;
        byte[] buffer = new byte[81];
        int nbytes;
        try {
            fis = new FileInputStream("leer.txt");
            nbytes = fis.read(buffer, 0, 81);
            String texto = new String(buffer, 0, nbytes);
            System.out.println(texto);
        } catch (IOException ioe) {
            System.out.println("Error: " + ioe.toString());
        } finally {
            try {
                if (fis != null) fis.close();
            } catch (IOException ioe) {
                System.out.println("Error al cerrar el archivo.");
            }
        }
    }
}
```

FileWriter

La clase FileWriter hereda de la clase Writer y permite escribir un flujo de caracteres en un archivo de texto plano.

BufferedWriter

La clase BufferedWriter hereda de la clase Writer y permite escribir un buffer para realizar una escritura eficiente de caracteres desde la aplicación hacia el archivo destino.

```java
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.io.IOException;

public class ClaseFileWriter{
    public static void main (String [] leer){
        String texto = "";
        try{
            BufferedReader br;
            br = new BufferedReader(new InputStreamReader(System.in));
            System.out.println("Escribir texto:");
            texto = br.readLine();
            FileWriter fw = new FileWriter("archivo.txt");
            BufferedWriter bw = new BufferedWriter(fw);
            PrintWriter salida = new PrintWriter(bw);
            salida.println(texto);
            salida.close();
        } catch (IOException ioe){
            System.out.println("\n\nError al abrir o guardar el archivo:");
            ioe.printStackTrace();
        } catch (Exception e){
            System.out.println("\n\nError al leer de teclado:");
            e.printStackTrace();
        }
    }
}
```

FileReader

La clase Reader se utiliza para obtener los caracteres ingresados desde una fuente. La clase FileReader hereda de Reader y permite leer flujos de caracteres de un archivo de texto plano.

BufferedReader

La clase BufferedReader crea un buffer para realizar una lectura de caracteres. Dispone del método readLine que permite leer una línea de texto.

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class ClaseFileReader{
    public static void main (String [] escribir){
        String texto = "";
        try {
            BufferedReader br;
            FileReader fr = new FileReader("leer.txt");
            br = new BufferedReader(fr);
            System.out.println("El texto contenido en el archivo leer.txt es:");
            String linea = br.readLine();
            while (linea != null ) {
                System.out.println(linea);
                linea = br.readLine();
            }
            br.close();
        } catch (IOException ioe){
            System.out.println("\n\nError al abrir o guardar el archivo:");
            ioe.printStackTrace();
        } catch (Exception e){
            System.out.println("\n\nError al leer de teclado:");
            e.printStackTrace();
        }
    }
}
```

Scanner

La clase Scanner permite leer flujos de bytes desde la entrada estándar.

El método next() obtiene el siguiente elemento del flujo de datos.

El método hasNext() verifica si el flujo de datos todavía posee elementos.

Nota: el delimitado de la clase Scanner por defecto es el espacio en blanco, aunque se puede cambiar utilizando el método useDelimitador().

```java
import java.util.Scanner;

public class EjScanner {

    public static void main(String [] args) {
        try {
            String cad = "";
            Scanner s = new Scanner(System.in);
            System.out.println(s.nextLine());
            s.close();
        } catch (Exception e){
            e.printStackTrace();
        }
    }
}
```

Serialización

La serialización es un mecanismo para guardar los objetos como una secuencia de bytes y poderlos reconstruir en un futuro cuando se necesiten, conservando su estado

Serialización

Serializar un objeto Date

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.util.Date;

public class SerializeDate {

    SerializeDate() {
        Date d = new Date ();
        System.out.println(d);
        try {
            FileOutputStream f = new FileOutputStream ("date.ser");
            ObjectOutputStream s = new ObjectOutputStream (f);
            s.writeObject (d);
            s.close ();
        } catch (IOException e) {
            e.printStackTrace ();
        }
    }

    public static void main (String args[]) {
        new SerializeDate();
    }
}
```

Serialización

Deserializar un objeto Date

```
import java.io.FileInputStream;
import java.io.ObjectInputStream;
import java.util.Date;

public class DeSerializeDate {

    DeSerializeDate () {
        Date d = null;
        try {
            FileInputStream f = new FileInputStream ("date.ser");
            ObjectInputStream s = new ObjectInputStream (f);
            d = (Date) s.readObject ();
            s.close ();
        } catch (Exception e) {
            e.printStackTrace ();
        }
        System.out.println( "Deserialized Date object from date.ser");
        System.out.println("Date: "+d);
    }

    public static void main (String args[]) {
        new DeSerializeDate();
    }
}
```

¿Dudas?

ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

ECYS | DTT
DESARROLLO DE TRANSFERENCIA TECNOLÓGICA

Nombre de la actividad:	Evaluación de Rendimiento
Cantidad de participantes:	22
Doy fe que esta actividad está planificada en dtt (Sí/No):	Sí

Hora de inicio:	9:27
Hora de fin:	9:37
Duración(min):	10 min

Participantes:

Diego Fernando Cortez Lopez
Listado de participantes