Excepción creacion

```
class MiExcepcion extends Exception{
  public String toString(){
   return "Mensaje de la exception
personalizada"
  }

}

Lanzar la excepción Personalizada

if manzanas>5
   throw new MiExcepcion()

//La excepcion deberia ser manejada con el try
catch o el metodo deberia declarar la
excepcion con el throws
```

Lectura Archivos
Buffered Reader

```
//Se usa try with resources para que
automaticamente se cierre el BufferedReader.

try(BufferedReader lector = new BufferedReader(new
FileReader("generoMusicales.txt"))){
String a;
while((a=lector.readLine())!= null){
                System.out.println(a)
            }


        }
        catch(FileNotFoundException e){
            System.out.println("Archivo no
encontrado "+ e);
        }
        catch(IOException e){
            System.out.println("Error de
entrada/salida"+e);
        }
```

Escritura de Archivos
Buffered Writer

```
//El filewriter recibe true en el atributo
append, cuando se quiere agregar
información,
//Si se desea crear un archivo nuevo y
sobrescribir el existente se debe omitir el
true
try(BufferedWriter w = new
BufferedWriter(new
FileWriter("peliculas.txt",true))){

w.write("kslkalsd"+"\n")


}
catch (IOException ex) {
            System.out.println("Excepcion
en metodo escribirArchivos"+ex);
        }
```

Lectura Archivos NIO

```
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

try{
List<String> lines =
Files.readAllLines(Paths.get("archivo.txt"));
}
catch(IOException e){
System.out.println(e)
}
for (String line: lines){
 System.out.println(line)
}
```

Observaciones: Por favor tener en cuenta que puede generar una IOException. Nótese que el readAllLines debe ser usado para archivos pequeños y devuelve una lista de String

aporte de: Angela Carrera

Serialización

Para serializar un objeto seguiremos la siguiente estructura:
1-La clase del objeto a serializar tiene que implementar la interfaz "Serializable"(ej:**Public class Student implements Serializable**).
2-Luego en la clase principal o un método vamos a abrir el flujo de datos, recordando que esto nos puede generar una excepción tipo IOException por lo que deberemos manejarla(ej:**try{FileOutputStream flujo = new FileOutputStream("dirección de memoria del archivo"**);
....
3-Luego crearemos un objeto tipo ObjectOutputStream para leer el flujo de datos que acabamos de crear (ej:**ObjectOutputStream archivo = new ObjectOutputStream(flujo);**
4-Ahora vamos a serializar un objeto previamente guardado en una variable con el método write()(ej: **Student alumno = new Student(...);**
archivo.write(alumno)//de esta manera habremos serializado a alumno en el archivo)
5-Por último cerraremos el flujo de datos con el método close()cuando terminemos de serializar los datos(ej:**archivo.close()**)

//Si queremos que un atributo no se serialice, usamos la palabra reservada transient.

Aporte de:Rubén Alexander Saltos Arce

Deserialización

1-La clase del objeto a serializar tiene que implementar la interfaz "Serializable"(ej:Public class Student implements Serializable).
2.- Se leerá el archivo serializado de la siguiente manera (se ha usado try with resources)

```
try(ObjectInputStream st= new
ObjectInputStream( new
FileInputStream("archivo.ser"))){

Materia m = (Materia) st.readObject

}

catch(IOException e){
 System.out.println(e)}
catch( ClassNotFoundException e){
System.out.println(e)}
```

Observaciones: El try with resources hace que no debamos cerrar los streams, el downcasting debe ser hecho de acuerdo al objeto guardado en el archivo serializado

aporte de: Angela Carrera

Sobrescritura equals

```
 public boolean equals(Object obj) {
        if (this == obj) {
            return true;
        }
        if (obj == null) {
            return false;
        }
        if (getClass() != obj.getClass())
            return false;

    Aeropuerto other = (Aeropuerto) obj;
    if (!Object.equals(this.nombre,
other.nombre)) {
            return false;
        }
        return true;
    }
```

Sobrescritura hashcode

```
@Override
    public int hashCode() {

        return Objects.hash(this.nombre);
    }
```

Ejemplo 2 con mas campos

```
@Override
    public int hashCode() {

        return
Objects.hash(this.nombre,this.matricula );
    }
```

Aporte de:Leonardo Ortiz

Sobrescritura compareTo

Ejemplo de compareto de clase estudiante para dos campos matricula y apellido suponiendo q matricula es int y apellido es String

```
public int compareTo(Estudiante e){
   int resultado =
Integer.compare(this.matricula,
e.matricula);
//Si son iguales
if resultado == 0{
  resultado =
this.apellido.compareTo(e.apellido);
}
return resultado;
}
```

Observaciones:
* debe implementar la clase Comparable<Clase> de esto dependerá que

tipo recibe el compareto
* Si queremos que el orden predeterminado sea de mayor a menor en lugar de mayor a menor multiplicamos el valor de la variable resultado por -1

Maps

Agregar al mapa

```
mapa.put(clave, valor)
//Solo lo agrega si la clave no existe
mapa.putIfAbsent(clave, valor)
```

Obtener Valor

```
//El tipo de la clave dependera de la
deficicion del mapa.
Map<String, Integer> m= new….
Integer o = m.get("clave")
```

Obtener Claves

```
Set<Integer> claves = m.keySet();
```

Recorrer Mapa

```
for (String clave: m.keySet()){
   System.out.println(clave, m.get(clave))
}
```

Threads

Crear una thread

```
public class Hilo implements Runnable{
    String nombre;
    public Hilo(String nombre){
      this.nombre= nombre
    }
    public void run(){
      //dentro de este método va lo que deseo que la thread haga
    }}
//La thread puede tener atributos como cualquier clase
```

Inicializar una thread

```
Thread t1 = new Thread(new Hilo("Hilito!"));
t1.start()
```

Dormir una thread

```
try{
   Thread.sleep(1000) //1 segundo
}
catch(InterruptedException e){
sout(e)}
```

Threads JavaFx

```
//Usado en conjunto con las threads

Platform.runLater(()-> {
//Aqui va lo que quiero actualizar de javafx dentro de una thread
});
```

Crear Formas JAVA FX

```
//width, height
Rectangle r = new Rectangle(40,20);
r.setLayoutX(100);
r.setLayoutY(200);
Circle c = new Circle(40);
c.setLayoutX(200);


//root es de tipo Pane
root.getChildren().addAll(r,c);
```

BorderPane

setCenter(Node n)
setLeft(Node n)
setBottom(Node n)
setTop(Node n)
setRight(Node n