

ESTRUCTURA DE DATOS I

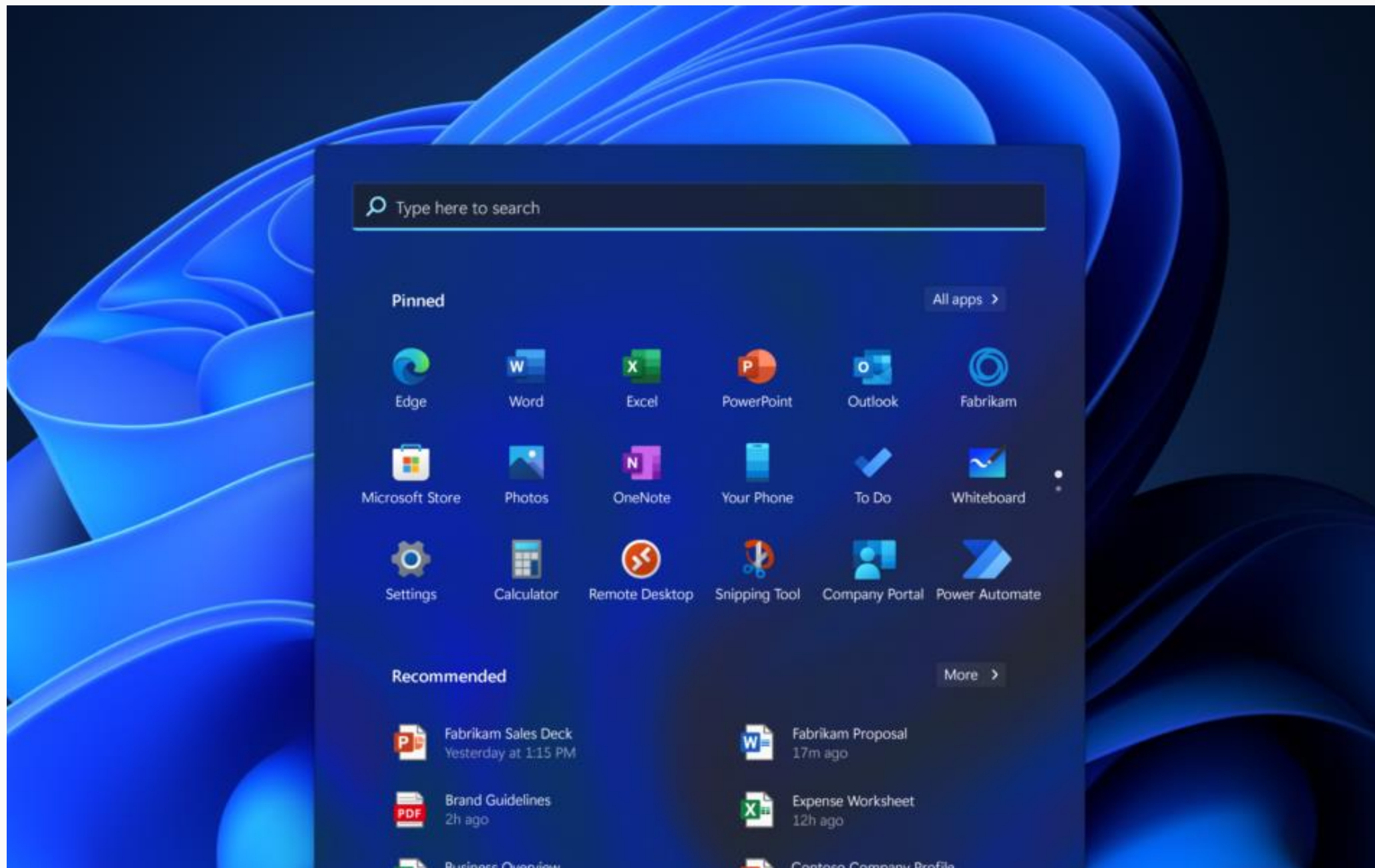
Control de Excepciones



MEng. Andrés Castillo

20/02/24

Control de excepciones



Control de excepciones

try-catch

- Dentro del bloque **try**, se ejecutan las sentencias sobre las que se quiere controlar las excepciones
- Dentro del bloque **catch**, se muestran las sentencias que se deben ejecutar en caso de error

```
try {  
    //SENTENCIAS A EJECUTAR  
} catch (Exception e) {  
    //SENTENCIAS QUE SE EJECUTAN SI HAY ERROR  
}
```

Como parámetro, el catch pone el tipo de excepción a controlar.
Si se quieren controlar todas, se usa Exception.

La sentencia finally es opcional, y contiene las sentencias que se van a ejecutar exista o no una excepción

Control de excepciones



¿Dónde?
¿Causa?
¿Rastro?



Control de excepciones



¿Dónde?
¿Causa?
¿Rastro?



Control de excepciones





```
public void reqFuncOpcion1( )
{
    String resultado = club.metodo1( );
}
```

```
public String metodo1( )
{
    Socio miSocio = (Socio)socios.get(0);
    miSocio.aumentarFondos(3.5);
    return "Aumento realizado !";
}
```



```
public void aumentarFondos( double pFondos ) throws Exception
{
    if( tipoSubscripcion == Tipo.VIP && pFondos + fondos > MONTO_MAXIMO_VIP )
    {
        throw new Exception( "Con este monto se excederían los fondos máximos" );
    }
    else if( tipoSubscripcion == Tipo.REGULAR && pFondos + fondos > MONTO_MAXIMO_REGULAR )
    {
        throw new Exception( "Con este monto se excederían los fondos máximos" );
    }
    else
    {
        fondos = fondos + pFondos;
    }
}
```

Recuperación

```
public void m1()  
{  
    try  
    {  
        obj1.m2();  
    }  
    catch (Exception e)  
    {  
        // Código para manejar el error atrapado  
    }  
}
```



```
public void m2() throws Exception  
{  
    obj2.m3();  
}
```



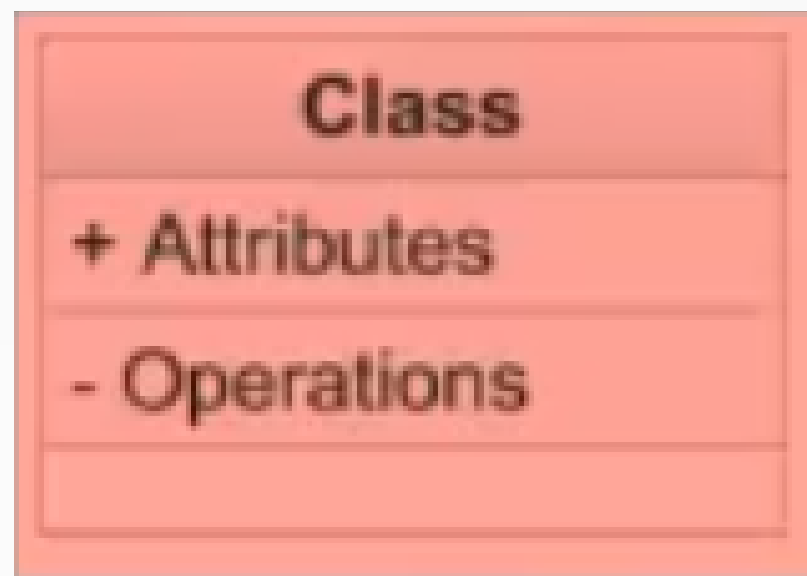
```
public void m3() throws Exception  
{  
    obj3.m4();  
}
```

Excepciones personalizadas

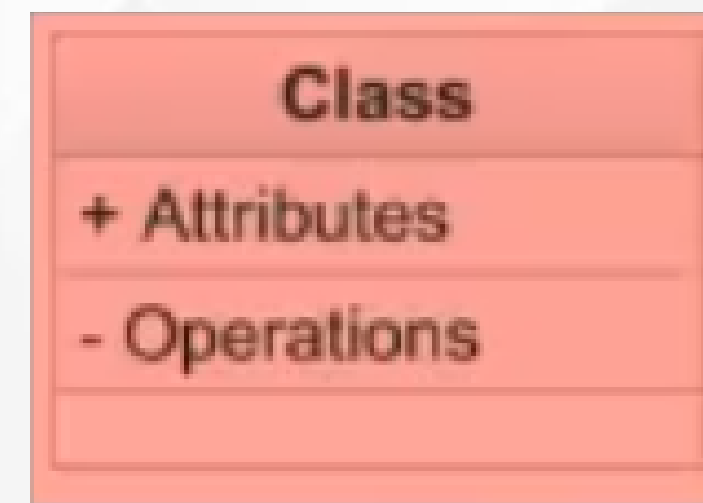


java.lang

Exception



Personal Exception




Extends



Excepciones personalizadas

```
public void leerEquipos() throws FileNotFoundException, IOException, ElementoExisteException
```

Invocar
método



```
public String metodo1( )
{
    try
    {
        leerEquipos();
        return "Equipos cargados correctamente";
    }
    catch (FileNotFoundException e)
    {
        return e.getMessage();
    }
    catch (IOException e)
    {
        return e.getMessage();
    }
    catch (ElementoExisteException e)
    {
        return e.getMessage();
    }
}
```

Control de excepciones

Tipo excepción

Descripción excepción

Pila ejecución

```
StudentException: Error finding students
    at StudentManager.findStudents(StudentManager.java:13)
    at StudentProgram.main(StudentProgram.java:9)
Caused by: DAOException: Error querying students from database
    at StudentDAO.list(StudentDAO.java:11)
    at StudentManager.findStudents(StudentManager.java:11)
    ... 1 more
Caused by: java.sql.SQLException: Syntax Error
    at DatabaseUtils.executeQuery(DatabaseUtils.java:5)
    at StudentDAO.list(StudentDAO.java:8)
    ... 2 more
```

A practicar!!

“La verdad solo se puede encontrar en un lugar: el código”.

Robert C. Martin

Ejemplo

RF1 - Generar un reporte de los discos más costosos, aquellos cuyo precio supere 5 mil.

Descripción:

En la disco-tienda se requiere un servicio que permita generar un reporte de los discos más costosos, es decir, cuyo precio supere los 5 mil.

Si no existen este tipo de discos registrados debe informarse al usuario del inconveniente.

Debe usarse una nueva excepción llamada ***DiscosCarosException***.

Reto

Persistencia por serialización - Numeral 5.8
páginas 153-159

Apropiación de JComboBox - numeral 5.9
páginas 159-160