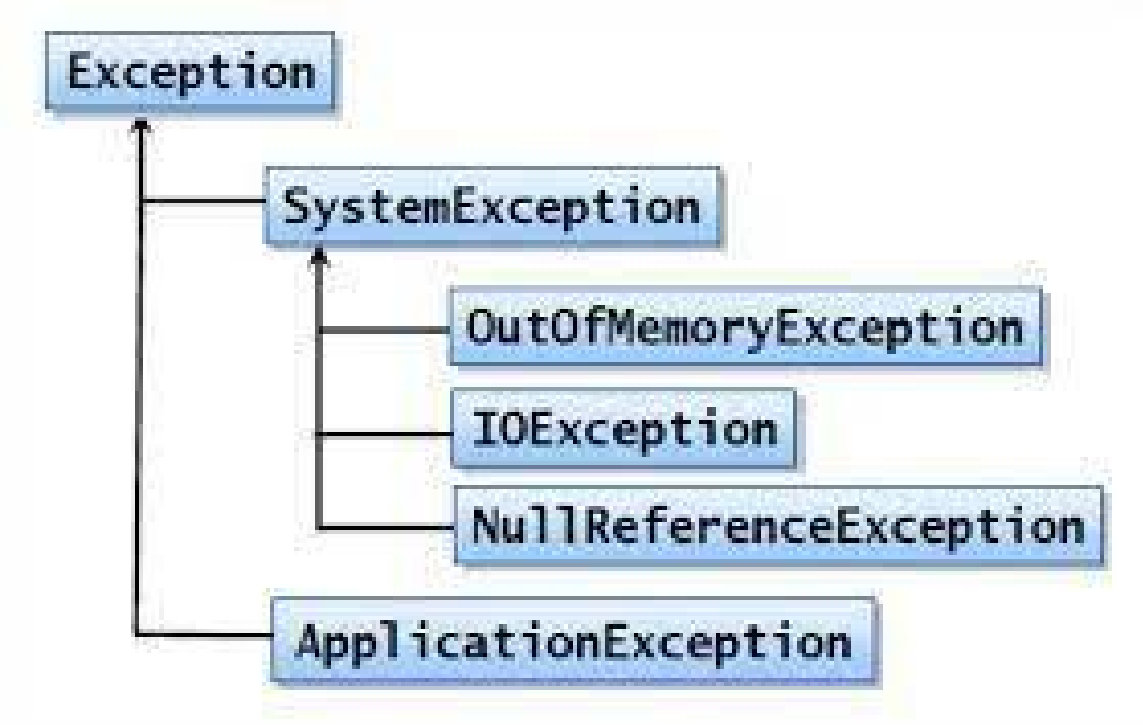
Nivel 9

Excepciones

¿Qué es una excepciones?

Una excepción es un aviso de error en el programa. Esto se produce cuando la ejecución de un método no termina correctamente. Cuando esto sucede, el programa puede terminar abruptamente su ejecución.

Cuando durante la ejecución de un método el ordenador detecta un error, crea un objeto de una clase especial para representarlo, el cual incluye toda la información del problema para que el mismo programa intente solucionarlo o que el usuario pueda ver lo que ha sucedido.



Ejemplo de Java.

Este ejemplo ilustra el caso en el cual durante la ejecución de un método se produce un error y el computador crea

un objeto para representarlo y permitir que en alguna parte del programa alguien lo atrape y lo use para evitar que el programa deje de funcionar.

```
public class C1
{
  private C2 atr;

  public void m1( )
  {
    atr.m2( );
  }
}
```

Supongamos que tenemos una clase C1, en la cual hay un método llamado m1(), que es llamado desde las clases de la interfaz del programa. Los objetos de la clase C1 tienen un atributo de la clase C2, llamado atr. Suponga además que dentro del método m1() se invoca el método m2() de la clase C2 sobre el atributo llamado atr.

```
public  class  C2
{
  public void m2( )
  {
    instr1;
    instr2;
    instr3;
  }
}
```

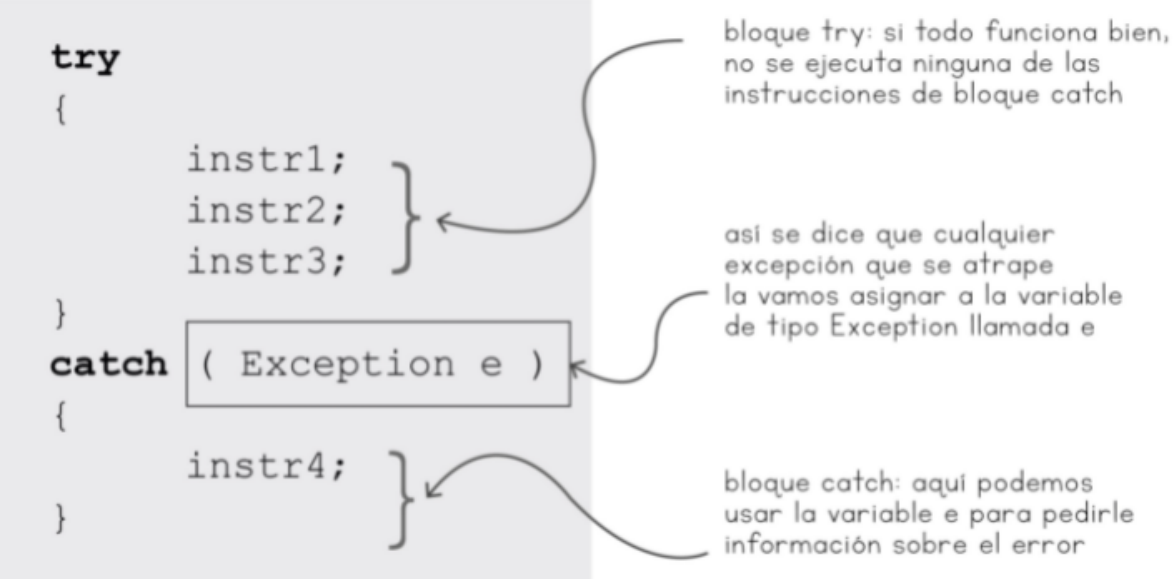
Dentro de la clase C2 hay un método llamado m2() que tiene 3 instrucciones, que aquí mostramos como instr1, instr2, instr3.

Dichas instrucciones pueden ser de cualquier tipo. Supongamos que se está ejecutando la instrucción instr2 del método m2() y se produce un error. En ese momento, a causa del problema el computador decide que no puede seguir con la ejecución del método (instr3 no se va a ejecutar).Crea entonces un objeto de la clase Exception que dice que el error sucedió en la instrucción instr2 del método m2() y explica la razón del problema. Luego, pasa dicho objeto al método m1() de la clase C1, que fue quien hizo la llamada. Si él lo atrapa el computador continúa la ejecución en el punto que dicho método indique. Si el método m1() no atrapa la excepción, este objeto pasa a la clase de la interfaz que hizo la llamada. Este proceso se repite hasta que alguien atrape la excepción o hasta que el programa completo se detenga. Entendemos por manejar una excepción el hecho de poderla identificar, atraparla antes de que el programa deje de funcionar y realizar una acción para recuperarse del error.

Instrucción try catch.

En la instrucción try-catch hay dos bloques de instrucciones, con los siguientes objetivos:

Delimitar la porción de código dentro de un método en el que necesitamos desviar el control si una excepción ocurre allí (la parte try). Si se dispara una excepción en alguna de las instrucciones del bloque try, la ejecución del programa pasa inmediatamente a las instrucciones del bloque catch. Si no se dispara ninguna excepción en las instrucciones del bloque try, la ejecución continúa después del bloque catch. Definir el código que manejará el error o atrapará la excepción.



Aplicación en Java.

```
public void ejemplo( String pCedula, String pNombre, Tipo pTipo )
{
  try
  {
    club.afiliarSocio( pCedula, pNombre, pTipo );
    totalSocios++;
  }
  catch( Exception e )
  {
    String ms = e.getMessage( );
    JOptionPane.showMessageDialog( this, ms );
  }
}
```

Si en la llamada del método afiliarSocio se produce una excepción, ésta es atrapada y la ejecución del programa continúa en la primera

instrucción del bloque catch. Notamos que en ese caso, la instrucción que incrementa el atributo totalSocios no se ejecuta.

La primera instrucción del bloque catch pide al objeto que representa la excepción el mensaje que explica el problema.

La segunda instrucción del bloque catch despliega una pequeña ventana de diálogo con el mensaje que traía el objeto e de la clase Exception. En este ejemplo, la intención es comunicarle al usuario que hubo un problema y que no se pudo realizar la afiliación del socio al club.

Construcción de un objeto Exception y la instrucción Throw.

Cuando necesitamos disparar una excepción dentro de un método utilizamos la instrucción throw del lenguaje Java. Esta instrucción recibe como parámetro un objeto de la clase Exception, el cual es lanzado o disparado al método que corresponda.

Aplicación en Java.

Este método lanza una excepción a aquél que lo llama, si le pasan como parámetro la información de un socio que ya existe o si el socio que se desea afiliar tiene suscripción VIP y ya se alcanzó el máximo número de suscripciones VIP que maneja el club.

El constructor de la clase Exception recibe como parámetro una cadena de caracteres que describe el problema detectado.

Cuando un método atrape esta excepción y le pida su mensaje (getMessage()), el objeto va a responder con el mensaje que le dieron en el constructor.

En este ejemplo, cuando se detecta el problema se crea el objeto que representa el error y se lo lanza, todo de una sola vez. Pero podríamos haber hecho lo mismo en dos instrucciones separadas.

La clase Exception es una clase de Java que ofrece múltiples servicios, que se pueden consultar en la documentación. Los más

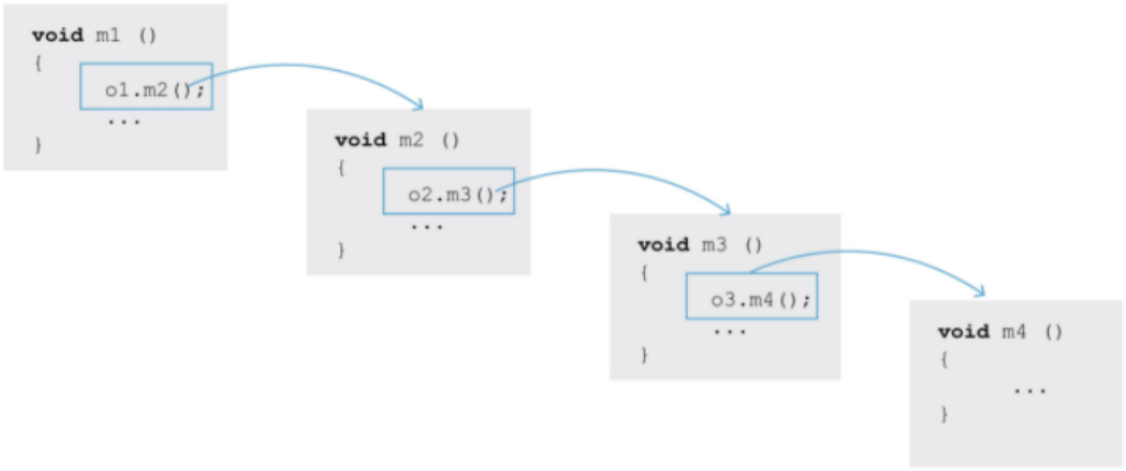
usados son getMessage(), que retorna el mensaje con el que fue creada la excepción, y printStackTrace(), que imprime en la consola de ejecución la traza incluida en el objeto (la secuencia anidada de invocaciones de métodos que dio lugar al error), tratando de informar al usuario respecto de la posición y la causa del error.

Si utilizamos las siguientes instrucciones después de atrapar la excepción del método afiliarSocio() en caso de que ya exista un socio con la misma cédula

```
public void afiliarSocio( String pCedula, String pNombre, Tipo pTipo ) throws Exception
{

  // En caso de que el tipo de suscripción del nuevo socio sea VIP, es necesario
  // revisar que no se haya alcanzado el límite de suscripciones VIP que maneja el club
  if( pTipo == Tipo.VIP && contarSociosVIP( ) == MAXIMO_VIP )
  {
    // Si ya se alcanzó el número máximo de suscripciones VIP, se lanza una excepción
    throw new Exception( "El club en el momento no acepta más socios VIP" );
  }

  // Revisar que no haya ya un socio con la misma cédula en el club
  Socio s = buscarSocio( pCedula );

  if( s == null )
  {
    // Se crea el objeto del nuevo socio (todavía no se ha agregado al club)
    Socio nuevoSocio = new Socio( pCedula, pNombre, pTipo );

    // Se agrega el nuevo socio al club
    socios.add( nuevoSocio );
  }
  else
  {
    // Si ya existía un socio con la misma cédula, se lanza una excepción
    throw new Exception( "El socio ya existe" );
  }
}
```

Recuperación de una situación anormal.

Supongamos ahora que durante la ejecución del método m4() se dispara una excepción. Es parte de nuestras decisiones de diseño decidir quién será el responsable de atraparla y manejarla. Una posibilidad es que el mismo método m4() la atrape y la procese. Otra posibilidad es que la responsabilidad se delegue hacia arriba, dejando que sea el método m3() o el método m2() o el método m1() quien se encargue de atrapar la excepción.



El método encargado de atrapar una excepción utiliza la instrucción try-catch, mientras que los métodos que sólo la dejan pasar lo declaran en su signatura (throws Exception).

Referencias.

1. Universidad de los Andes. (2019). *Manejo de las Excepciones · Fundamentos de Programación*. Gitbooks.io. https://universidad-de-los-andes.gitbooks.io/fundamentos-de-programacion/content/Nivel4/5_ManejoDeLasExcepciones.html