Excepciones

Facultad de Ciencias
Diana Isabel Ramírez García

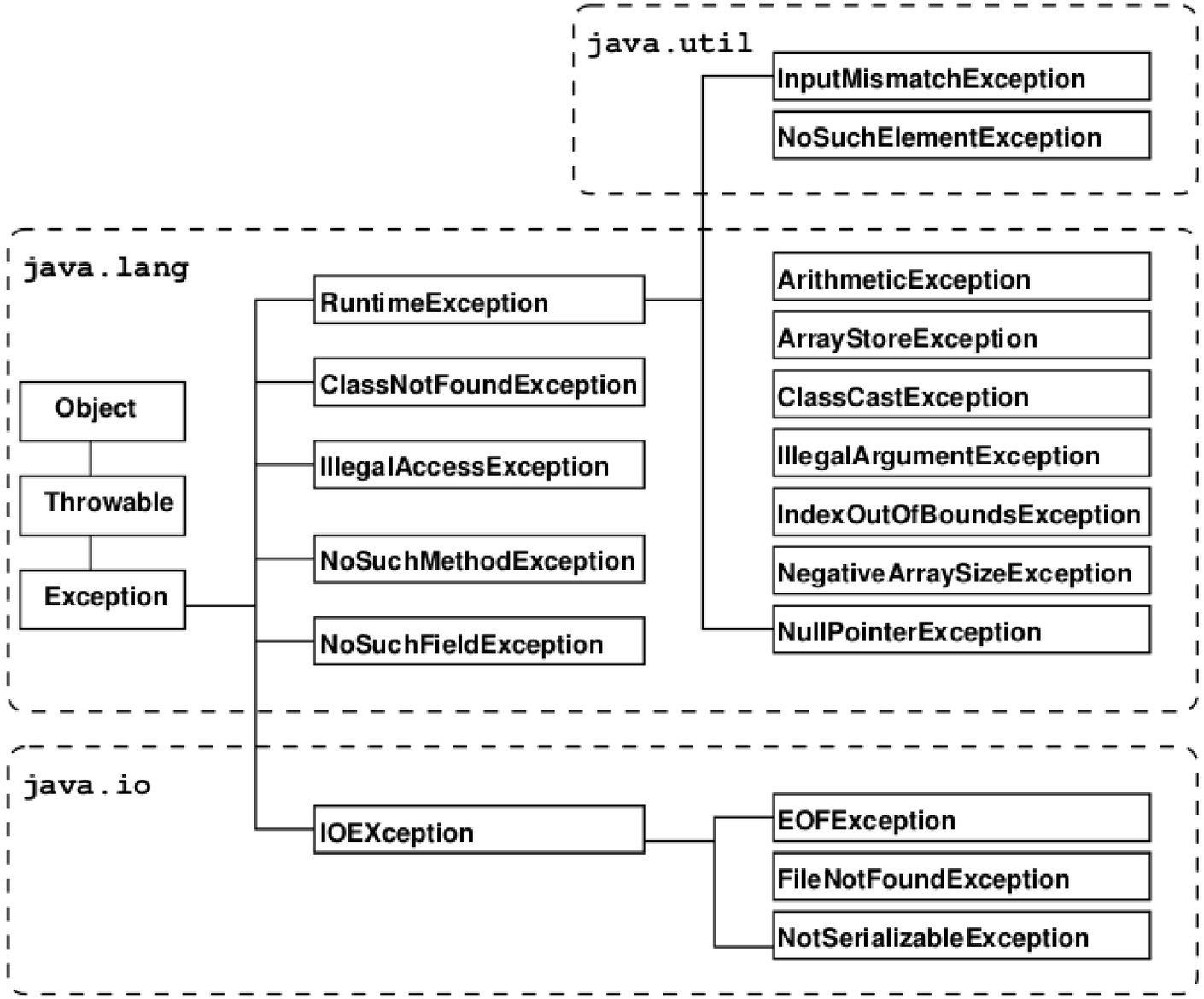
Introducción a Ciencias de la Computación 2021-1

¿Qué es una Excepción?

Una excepción es un evento que ocurre en cualquier momento de ejecución de un programa y que modifica el flujo normal de éste.

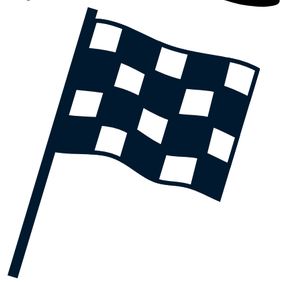
- Una excepción es un objeto de la **Exception**
- Almacenan información que se manda cuando surge una anomalía en la ejecución

Jerarquía de excepciones



Estados de una Excepción

- **Disparo:** Surge una anomalía, entonces se crea una excepción con:
  - Tipo de excepción
  - Estado del programa
- **Atrapado:** El sistema de ejecución busca instrucciones a realizar cuando sucede la excepción.
- **Terminación:** Si no se maneja la excepción se termina la ejecución. Por el otro lado es posible atrapar la excepción y continuar la ejecución.



Disparo de una Excepción

- Para disparar una excepción utilizamos **throw** con algún objeto de **Exception.**
- Si se ejecuta la instrucción es porque sucedió un error.

```
throw new IllegalArgumentException("El monto no es correcto");
```

Disparo de una Excepción

- Para disparar una excepción utilizamos **throw** con algún objeto de **Exception.**
- Si se ejecuta la instrucción es porque sucedió un error.

  throw new IllegalArgumentException("El monto no es correcto");
- **Palabra reservada para indicar que se dispara la excepción**

Disparo de una Excepción

- Para disparar una excepción utilizamos **throw** con algún objeto de **Exception.**
- Si se ejecuta la instrucción es porque sucedió un error.

  throw **new IllegalArgumentException("El monto no es correcto")**;
- **Objeto de la clase Exception**

Disparo de una Excepción

- Para disparar una excepción utilizamos **throw** con algún objeto de **Exception.**
- Si se ejecuta la instrucción es porque sucedió un error.

  throw new **IllegalArgumentException**("El monto no es correcto");
- **Tipo de excepción que se lanza**

Disparo de una Excepción

- Para disparar una excepción utilizamos **throw** con algún objeto de **Exception.**
- Si se ejecuta la instrucción es porque sucedió un error.

  throw new IllegalArgumentException("**El monto no es correcto**");
- **Mensaje que se muestra cuando sucede la excepción**

Disparo de una Excepción

- Si puede suceder en el método una excepción que no pertenece a **RuntimeException** entonces se debe incluir en la firma del método **throws con el nombre de la excepción** además de lanzar la excepción

```
public double retirar(double monto) throws Exception {}
```

Disparo de una Excepción

- Si puede suceder en el método una excepción que no pertenece a **RuntimeException** entonces se debe incluir en la firma del método **throws con el nombre de la excepción** además de lanzar la excepción

  public double retirar(double monto) **throws** Exception {}
- **Palabra reservada para advertir que puede suceder una excepción y para tomar las acciones necesarias si llega a disparar la excepción**

Disparo de una Excepción

- Si puede suceder en el método una excepción que no pertenece a **RuntimeException** entonces se debe incluir en la firma del método **throws con el nombre de la excepción** además de lanzar la excepción

  public double retirar(double monto) throws **Exception** {}
- **Nombre de la excepción a lanzar, en este caso es uno de la clase padre Exception.**

Disparo de una Excepción

```
public double retirar(double monto) throws Exception {
    if (monto <= 0) {      //Como la excepción no pertenece a RunTimeException ponemos throws
        throw new Exception("La cantidad a retirar debe ser positiva");
    }                      //Utilizamos throw para lanzar la excepción
    if (saldo < monto) {
        saldo -= 500;
        throw new Exception("Fondos insuficientes para realizar el retiro");
    }                      //Podemos lanzar más de una excepción
    saldo -= monto;
    return monto;          //Si se lanza una excepción ya no se ejecuta el return  ya que throw
                           es equivalente y en su lugar se manda la excepción
}
```

Disparo de una Excepción

- Un método puede lanzar más de un tipo de excepción, en ese caso es necesario utilizar throws e incluir los nombres de las excepciones separadas por comas.
- Dentro del método debera haber al menos un throw por cada tipo de excepción.

Manejo de una Excepción

```
try {
instrucciones
}
catch (Excepción1 e 1 ) {
instrucciones
}
...
catch (Excepción n e n ) {
instrucciones
}
finally {
instrucciones
}
```

- Para atrapar una excepción es necesario un manejador de excepciones, en este caso la instrucción try.
- Este se utiliza **comúnmente dentro del main** , ya que al momento de ejecución debe atrapar la excepción

Manejo de una Excepción

```
try {
instrucciones
}
catch (Excepción1 e 1 ) {
instrucciones
}
...
catch (Excepción n e n ) {
instrucciones
}
finally {
instrucciones
}
```

- **Dentro del try se escriben las instrucciones que pueden disparar la(s) excepcion(es).**
- **La declaración de variable se realiza fuera de try debido al alcance.**

Manejo de una Excepción

```
try {
instrucciones
}
catch (Excepción_1 e1 ) {
instrucciones
}
...
catch (Excepción_n en ) {
instrucciones
}
finally {
instrucciones
}
```

- **Catch como parámetro tiene un objeto Exception.**
- **Es posible tener más de un catch para atrapar distintas excepciones y manejarlas de distinta manera.**

Fin de una Excepción

```
try {
instrucciones
}
catch (Excepción_1  e1 ) {
instrucciones
}
...
catch (Excepción_n en ) {
instrucciones
}
finally {
instrucciones
}
```

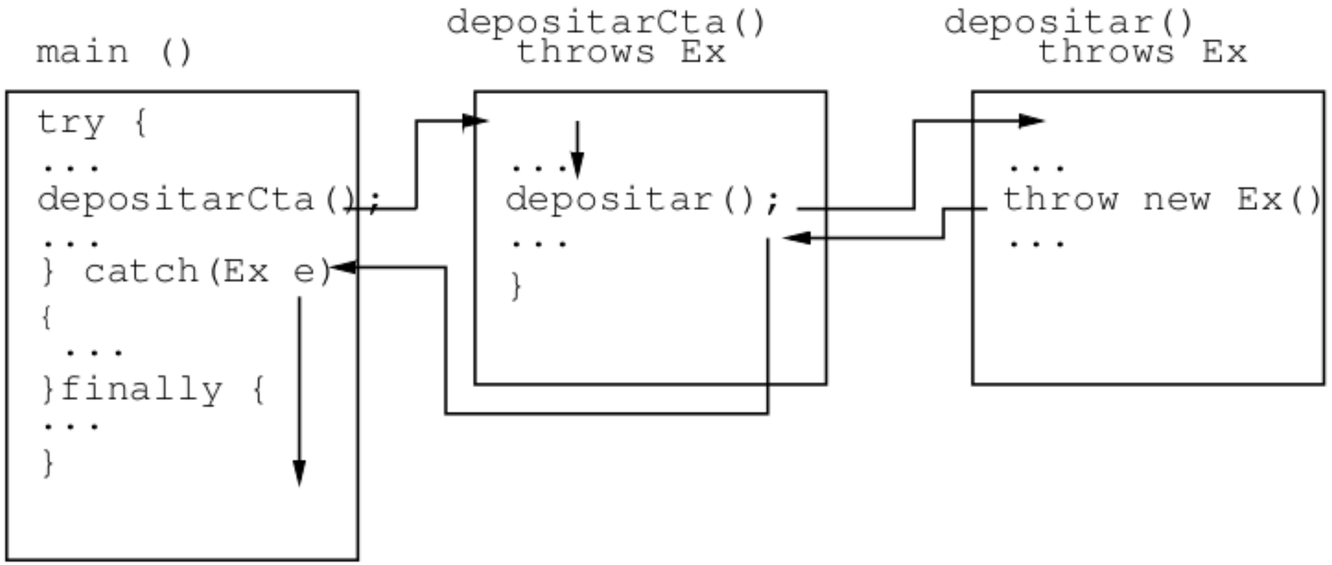
- **Dentro de finally se ponen las instrucciones para establecer un comportamiento adecuado para continuar la ejecución.**
- **finally es opcional.**
- **Suceda o no una excepción finally siempre se ejecutará.**

Creación de excepciones propias

- Es posible extender la clase Exception para crear excepciones propias.
- A la subclase que se cree no se le añade comportamiento alguno.
- Creamos excepciones propias para mejorar la legibilidad del código.

Propagación de excepciones

Si una excepción no se trata en el método que se originó, se propaga al método que lo llamó y así sucesivamente hasta encontrar un manejador de excepciones o llegar al entorno de Java y terminar abruptamente.



Recuperación de excepciones

- Podemos escribir la instrucción try dentro de un ciclo, para corregir el error y así sucesivamente hasta que sea correcto.
- La instrucción try se refiera todos los bloques(try,catch y finally)

Ventajas

- Podemos separar código para el manejo de error del código normal.
- Podemos agrupar y diferenciar errores.
- Podemos crear excepciones propias que se manejan de la misma manera.

Bibliografía

- López, A.. (2014). Introducción al desarrollo de programas con Java. México: Las prensas de Ciencias.