Manejo de Excepciones

Profesor: Andrés Guzmán F

¿Qué son las excepciones?

Una excepción es un problema o evento que ocurre durante la ejecución del programa que interrumpe el flujo normal.

Características

- Separa el código que gestiona los errores del código principal del programa
- Nos permita manejar el error y continuar con la ejecución del programa
- Agrupar y diferenciar entre diferentes tipos de errores
- Propagar errores hacia arriba en la pila de llamadas (StackTrace)

Sintaxis
try...catch...finally

```
try {
   [ bloque que lanza la excepción ...]
} cacth (Exeption ex) {
    [ acá manejamos el error ...]
} finally {
     [ bloque opcional, siempre se ejecuta ...]
}
```

Una estructura que nos permite capturar excepciones, reaccionar a un error de ejecución, podemos imprimir mensajes de error "a la medida" y continuar con la ejecución del programa.

Capturar múltiples excepciones

try...catch...finally

```
try {
  String valor = JOptionPane.showInputDialog(null, "Ingresar un entero:");

  // Un valor no numérico lanzará un NumberFormatException
  int divisor = Integer.parseInt(valor);

  // Si la división es 0, esto resultará un ArithmetricException
  System.out.println(10 / divisor);

} cacth (NumberFormatException nfe) {
   [ acá manejamos el error NumberFormatException ...]
} cacth (ArithmetricException ae) {
    [acá manejamos el error ArithmetricException ...]
}
```

La clase Exception

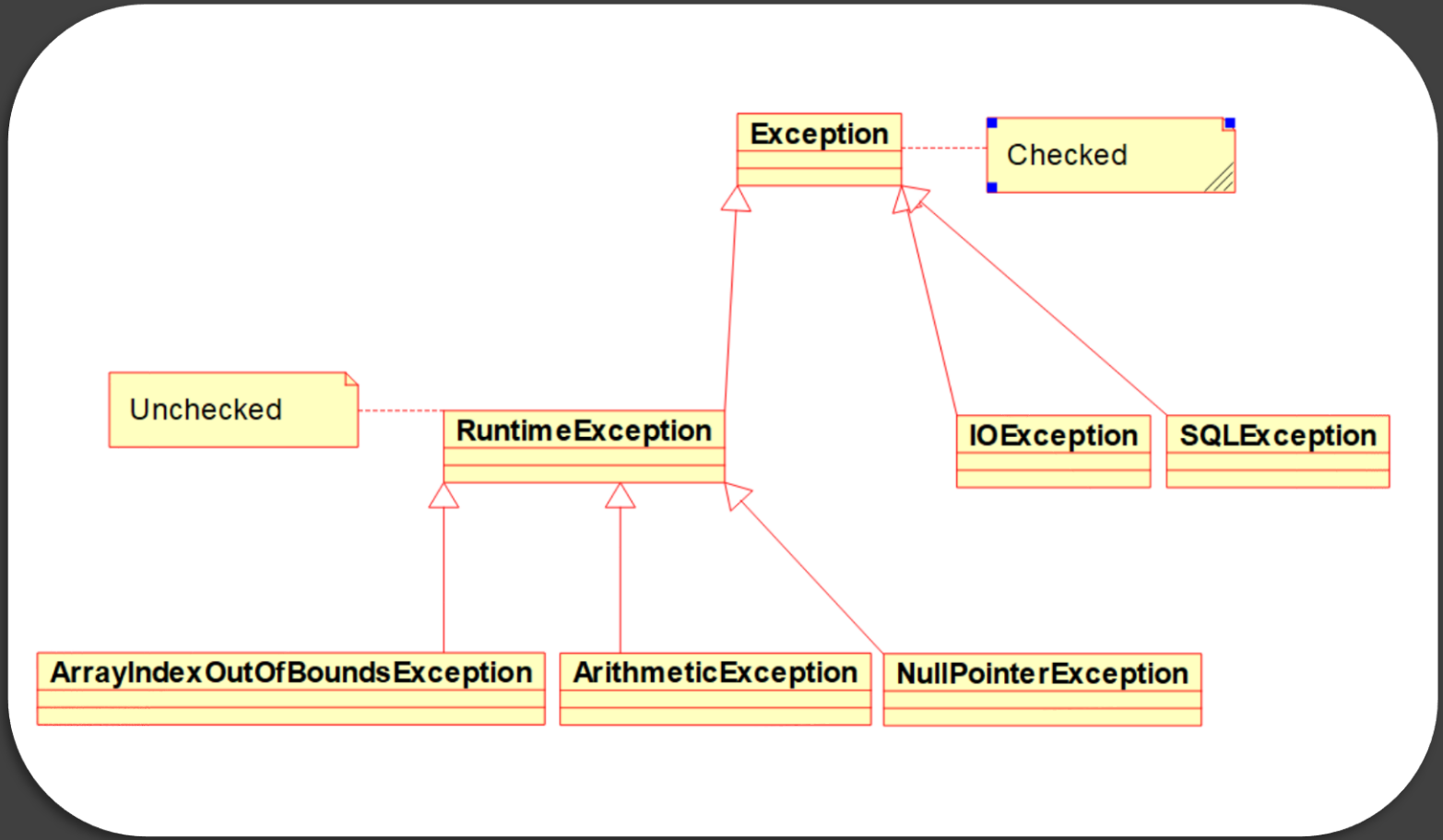
- Cuando se lanza una excepción, lo que se hace es lanzar una instancia de Exception o de una clase derivada.
- Esta clase tiene dos constructores y dos métodos importantes:

```
Exception e = new Exception();
```

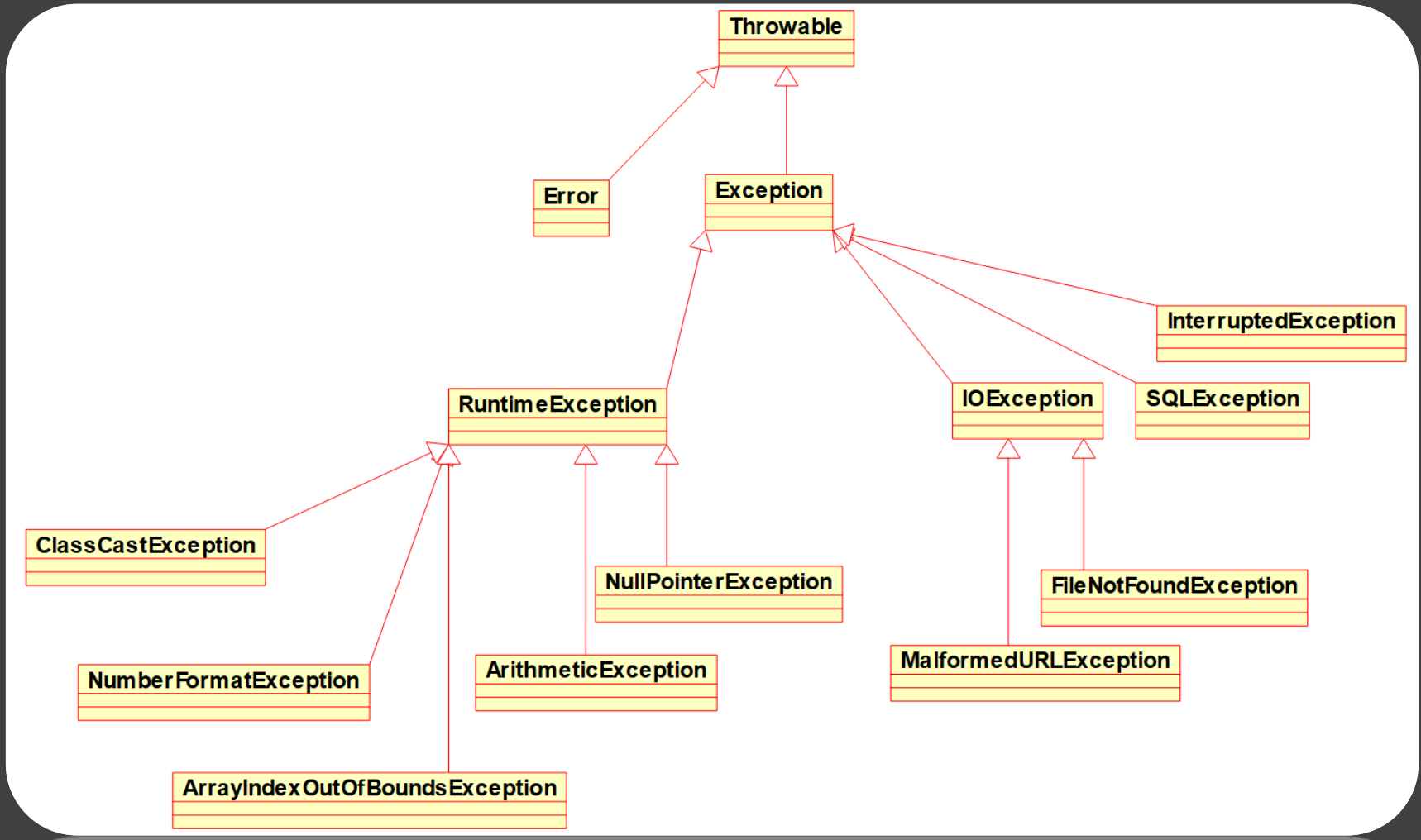
```
String mensaje = "algún mensaje de error";
Exception e = new Exception (mensaje);
```

```
String mensaje = e.getMessage();
e.printStackTrace();
```

Existen 2 tipos de Excepciones
Chequeadas y NO chequeadas



Existen 2 tipos de Excepciones Chequeadas y NO chequeadas



La Clausula throws

```
public class ClienteRepositorio {
    public Cliente porId(int id) throws Exception {
        [...]
    }
}
```

```
public static void main(String[] args) throws Exception {
        ClienteRepositorio repo = new ClienteRepositorio();
        Cliente cliente = repo.porId(2);
    }
```

- Los métodos deben capturar o propagar todas las excepciones chequeadas que puedan ser lanzadas dentro de su ámbito
- Esto es con la clausula throws que lista una o varias excepciones que son lanzadas en el método.

La sentencia throw

```
public class ClienteRepositorio {

   public Cliente porId(int id) throws Exception {
      if(id == 0){
         throw new Exception("id no puede ser cero");
      }
      [...]

   }

}
```

```
public static void main(String[] args) throws Exception {

      ClienteRepositorio repo = new ClienteRepositorio();

      Cliente cliente = repo.porId(2);

   }
```

- Los métodos utilizan la sentencia throw para lanzar una excepción
- Esta sentencia requiere un sólo argumento, un objeto Throwable

Crear clases de Excepciones

```
public class PersonalizadaException extends Exception {
    public PersonalizadaException(String mensaje) {
        super(mensaje);
    }
}
```

```
public class ClienteRepositorio {
    public Cliente porId(int id) throws PersonalizadaException {
        if(id == 0){
            throw new PersonalizadaException("id no puede ser cero");
        }
    }
}
```

Herencia de Excepciones

```
public class NoZeroIdException extends PersonalizadaException {
    public NoZeroException(String mensaje) {
        super(mensaje);
    }
}
```

```
public class PersonalizadaException extends Exception {
    public PersonalizadaException(String mensaje) {
        super(mensaje);
    }
}
```

```
public class ClienteRepositorio {
    public Cliente porId(int id) throws PersonalizadaException {
        if(id == 0){
            throw new NoZeroIdException("id no puede ser cero");
        }
    }
}
```

Sobreescritura y Excepciones

```
public class ClienteRepositorio implements CrudRepositorio {

    public Cliente porId(int id) throws PersonalizadaException {
        if(id == 0){
            throw new NoZeroIdException("id no puede ser cero");
        }
    }

}
```

```
public interface CrudRepositorio {

    Cliente porId(int id) throws PersonalizadaException;

}
```

```
public class ClienteRepositorio implements CrudRepositorio {

    public Cliente porId(int id) throws NoZeroIdException {
        if(id == 0){
            throw new NoZeroIdException("id no puede ser cero");
        }
    }

}
```