Crear excepciones personalizadas

DigitalHouse >
Coding School

Certified Tech Developer
The Ultimate Degree

Índice

1 | La clase fecha

La clase que queremos proteger

Vamos a tomar nuevamente la clase fecha, pero queremos diferenciar el tipo de error que ocurrió. Tenemos dos posibles errores, que el día esté fuera de rango o que el mes esté fuera de rango. Por fines prácticos tomamos días válidos de 1 a 31.

Fecha
- int day - int month - int year
+ Fecha(int d, int m, int y)

Usamos los nombres de atributos en inglés para no tener que usar la “**ñ**”.

2 | Extender Exception

Extender Exception

```
public class FechaException extends  Exception{

   public FechaException(){
      super();
   }
   public FechaException(String mensaje){
      super(mensaje);
   }
   public String toString(){
      return "Se produjo la siguiente Excepción "+ this.getClass().getName() +"\n" +
               " Mensaje: " + this.getMessage() + "\n" ;
   }}
```

Extendemos Exception y creamos dos constructores: uno por defecto que no tiene parámetros y el otro con parámetros y sobreescribimos el toString(). En el constructor con parámetros puedo recibir un mensaje que es el que luego me va a mostrar el error en detalle.

Extender Exception

En este ejemplo extendemos de Exception, pero se puede extender de cualquier Excepción definida en el API de Java. Siempre es conveniente utilizar la más relacionada con la condición que se quiere proteger.

Para crear nuestras propias excepciones, tenemos que heredar de la excepción que esté más relacionada con la condición a proteger

3 Usar nuestra propia excepción

Usar nuestras propias excepciones

```
class Fecha{
     private int day;
     private int month;
     private int year;

     public Fecha(int d, int m, int y){
          day=d;
          month=m;
          year=y;
     }
}
```

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

     Fecha fecha= new Fecha(100,-100,1000);
}
```

La clase fecha sin proteger la integridad de los datos.

```
class Fecha{
     private int day;
     private int month;
     private int year;

     public Fecha(int d, int m, int y) throws FechaException{
          if (d<1||d>31)
               throw new FechaException("Error en el día");
          day=d;
          if (m<1||m>12)
               throw new FechaException("Error en el mes");
          month=m;
          year=y;
     }}
```

Veamos en detalle, qué ocurre en el método.

```
public Fecha(int d, int m, int y) throws FechaException{
     if (d<1||d>31)
          throw new FechaException("Error en el día");
     day=d;
     if (m<1||m>12)
          throw new FechaException("Error en el mes");
     month=m;
     year=y;
}}
```

El método puede lanzar una excepción de tipo FechaException. Si el día está fuera de rango, se lanza la excepción con el mensaje que informa error en el día. Si el mes está fuera de rango, se lanza la excepción con el mensaje que informa error en el mes. Si se genera la excepción, el código no se sigue ejecutando.

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

     try{
          Fecha fecha= new Fecha(-1,10,2000);}
     catch(FechaException excepcion){
          System.err.println(exception.getMessage());
     }

}
```

Error en el día

Obtendremos el mensaje que programamos en nuestra clase, si cometemos error en el mes, el mensaje lo indicaría, ya que generamos la excepción indicando qué error ocurrió.


DigitalHouse>
Coding School