Criando exceções personalizadas

Certified Developer
The Ultimate Tech Degree

DigitalHouse >
Coding School

Temas

1 A Classe Data

2 Extender Exception

3 Usando nossa própria exceção

1 | A Classe Data

A classe que queremos proteger

Faremos a classe Data novamente, mas queremos diferenciar o tipo de erro que ocorreu. Temos dois erros possíveis: o dia está fora do intervalo, ou que o mês está fora do intervalo. Para fins práticos, consideramos dias válidos de 1 a 31.

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

2 | Extender Exception

Extender Exception

Estendemos Exception e criamos dois construtores: um por padrão que não tem parâmetros e o outro com parâmetros e sobrescrevemos o toString (). No construtor com parâmetros posso receber uma mensagem que é a que me mostra o erro detalhadamente.

```
public class DataException extends  Exception{

   public DataException(){
      super();
   }
   public DataException(String mensagem){
      super(mensagem);
   }
   public String toString(){
      return "A seguinte exceção ocorreu "+ this.getClass().getName() +"\n" +
                " Mensagem: " + this.getMessage() + "\n" ;
   }}
```

Extender Exception

Neste exemplo, estendemos de Exception, mas pode estender-se de qualquer Exception definida na API Java. É sempre aconselhável usar aquele que estiver mais relacionado com a condição que deseja proteger.

Para criar nossas próprias exceções, temos que herdar da exceção que está mais relacionada à condição para proteger.

3 | Usando nossa própria exceção

Usando nossas próprias exceções

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

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

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

      Data data= new Data(100,-100,1000);
}
```

A classe Data sem proteger a integridade dos dados.

Vamos ver em detalhes o que acontece no método.

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

     public Data(int d, int m, int y) throws DataException{
          if (d<1||d>31)
               throw new DataException("Error no dia");
          day=d;
          if (m<1||m>12)
               throw new DataException("Error no mes");
          month=m;
          year=y;
     }}
```

O método pode lançar uma exceção do tipo DataException. Se o dia estiver fora do intervalo, a exceção será lançada com a mensagem de erro de relatório no dia. Se o mês estiver fora do intervalo, a exceção será lançada com a mensagem de erro de relatório do mês. Se a exceção for lançada, o código não continuará a ser executado.

```
public Data(int d, int m, int y) throws DataException{
     if (d<1||d>31)
          throw new DataException("Error no dia");
     day=d;
     if (m<1||m>12)
          throw new DataException("Error no mes");
     month=m;
     year=y;
}
```

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

     try{
          Data dataa= new Data(-1,10,2000);}
     catch(DataaException excepcion){
          System.err.println(exception.getMessage());
     }

}
```

Erro no dia

Iremos receber a mensagem que programamos na nossa classe, se errarmos no mês, a mensagem indicaria, pois geramos a exceção indicando qual erro ocorreu.


DigitalHouse>
Coding School