

## 86 - Excepciones propias

[Listado completo de tutoriales](#)

La librería o API de Java proporciona gran cantidad de clases que manejan casi cualquier tipo de excepción, pero no estamos obligados a utilizar solo esas clases sino que podemos crear nuestras propias excepciones.

Si queremos crear una excepción no verificada debemos heredar de `RuntimeException` y si queremos que sea verificada deberemos heredar de `Exception` o de alguna de sus subclases.

Veamos con un ejemplo como crear una clase que herede de `Exception`, luego en otra clase lanzar una excepción de la nueva clase creada y finalmente llamar al método que lanza la excepción.

### Problema:

Confeccionar una clase que administre una lista tipo pila (se debe poder insertar, extraer e imprimir los datos de la pila).

En el método extraer lanzar una excepción si la pila se encuentra vacía, crear una excepción propia que herede de `Exception`.

### Clase: PilaVacíaException

```
public class PilaVacíaException extends Exception {  
  
    public PilaVacíaException(String mensaje) {  
        super("Problema:" + mensaje);  
    }  
  
}
```

La clase 'PilaVacíaException' hereda de la clase 'Exception', esto significa que quien lance una excepción de este tipo luego deberá ser capturada en forma obligatoria.

### Clase: Pila

```
public class Pila {  
  
    class Nodo {  
        int info;  
        Nodo sig;  
    }  
  
}
```

```

private Nodo raiz;

public Pila() {
    raiz = null;
}

public void insertar(int x) {
    Nodo nuevo;
    nuevo = new Nodo();
    nuevo.info = x;
    if (raiz == null) {
        nuevo.sig = null;
        raiz = nuevo;
    } else {
        nuevo.sig = raiz;
        raiz = nuevo;
    }
}

public int extraer() throws PilaVacíaException {

```

El método 'extraer' lanza una excepción de tipo 'PilaVacíaException' en caso que la pila se encuentre vacía:

```

public int extraer() throws PilaVacíaException {
    if (raiz != null) {
        int informacion = raiz.info;
        raiz = raiz.sig;
        return informacion;
    } else {
        throw new PilaVacíaException("No hay mas elementos en la p
    }
}

```

En la main donde creamos un objeto de la clase 'Pila' insertamos 3 enteros:

```

Pila pila1 = new Pila();
pila1.insertar(10);
pila1.insertar(40);
pila1.insertar(3);

```

Seguidamente dentro de un bloque try/catch atrapamos si ocurre una excepción de tipo 'PilaVacíaException', esto ocurre cuando finaliza el while y volvemos a llamar al método extraer, siendo que la pila se encuentra vacía:

```

try {
    while (!pila1.vacia())
        System.out.println("Extraemos de la pila:" + pila1.extraer());
    System.out.println("Extraemos de la pila:" + pila1.extraer());
} catch (PilaVacíaException e) {
    System.out.println("La pila está vacía");
}

```

```

    } catch (PilaVacíaException ex) {
        System.out.println(ex.getMessage());
    }
}

```

Tenemos una salida similar a esta:

The screenshot shows an IDE with two files: `*PilaVacíaException.java` and `*Pila.java`. The first file defines a custom exception class. The second file implements a stack and demonstrates the exception. The console output shows the stack's state and the exception message when it becomes empty.

```

// *PilaVacíaException.java
1 public class PilaVacíaException extends Exception {
2
3     public PilaVacíaException(String mensaje) {
4         super("Problema:" + mensaje);
5     }
6 }
7

// *Pila.java
30
31 public int extraer() throws PilaVacíaException {
32     if (raiz != null) {
33         int informacion = raiz.info;
34         raiz = raiz.sig;
35         return informacion;
36     } else {
37         throw new PilaVacíaException("No hay mas elementos en la pila");
38     }
39 }
40
41
42 public static void main(String[] ar) {
43     Pila pila1 = new Pila();
44     pila1.insertar(10);
45     pila1.insertar(40);
46     pila1.insertar(3);
47     try {
48         while (!pila1.vacia())
49             System.out.println("Extraemos de la pila:" + pila1.extraer());
50         System.out.println("Extraemos de la pila:" + pila1.extraer());
51     } catch (PilaVacíaException ex) {
52         System.out.println(ex.getMessage());
53     }
54 }
55 }

// Console Output
<terminated> Pila [Java Application] C:\Program Files\Java\jdk-11.0.2\bin\javaw.exe (20 mar. 2019 22:01:55)
Extraemos de la pila:3
Extraemos de la pila:40
Extraemos de la pila:10
Problema:No hay mas elementos en la pila

```

## Problema propuesto

1. Plantear una clase llamada 'Termometro' que defina un atributo llamado temperatura. Lanzar una excepción propia llamada 'TemperaturaFueraRangoException' si se intenta fijar una temperatura con un valor menor a -192 o superior a 100.

[Solución](#)

[Retornar](#)