Sesión 21: Excepciones

Programación 2

Ángel Herranz

Abril 2019

Universidad Politécnica de Madrid

En capítulos anteriores

- 👍 Tema 1: Clases y Objetos
- 👍 Tema 2: Colecciones acotadas de Objetos
- 👍 Tema 4: Tipos Abstractos de Datos
- 👍 Tema 3: Programación Modular
- 👍 Tema 5: Herencia y Polimorfismo
- 🕒 Tema 6: Excepciones
- 🕒 Tema 7: Implementación de TADs lineales

`Nodo<T>`

En el capítulo de hoy

- 👍 Tema 6: Excepciones
  - Lo aplicamos a pilas acotadas
- ¿🕘? Tema 7: Implementación de TADs lineales
  - Pilas no acotadas redimensionando arrays

Pilas acotadas

- API: crear, apilar, desapilar, cima, está vacía y está llena
- Vamos a aplicar mucho de lo que hemos aprendido
- `PilaAcotada`: interfaz genérico
- `PilaAcotadaVacia`: excepción para cima y desapilar
- `PilaAcotadaLlena`: excepción para apilar
- `PilaArray`: implementación de `PilaAcotada`
- `TestPilaArray`: tests para `PilaAcotada`

Introducir excepciones

1. Analizar el código de apoyo en las siguientes transparencias
2. Elevar y declarar las excepciones
3. Capturar las excepciones en el test

Código de apoyo

PilaAcotada

```
public interface PilaAcotada<T> {
  void apilar(T dato) throws PilaAcotadaLlena;
  T cima() throws PilaAcotadaVacia;
  void desapilar() throws PilaAcotadaVacia;
  boolean llena();
  boolean vacia();
}
```

PilaAcotadaLlena

```
public class PilaAcotadaLlena
  extends Exception
{
  public PilaAcotadaLlena() {
  }
}
```

PilaAcotadaVacia

```
public class PilaAcotadaVacia
  extends Exception
{
  public PilaAcotadaVacia() {
  }
}
```

PilaArray i

```
public class PilaArray<T>
  implements PilaAcotada<T>
{
  private T[] datos;
  private int cima;

  @SuppressWarnings("unchecked")
  public PilaArray(int capacidad) {
    datos = (T[]) new Object[capacidad];
    cima = -1;
  }
```

PilaArray ii

```
public void apilar(T dato) throws PilaAcotadaLlena {
  cima++;
  datos[cima] = dato;
}

public T cima() throws PilaAcotadaVacia {
  return datos[cima];
}

public void desapilar() throws PilaAcotadaVacia {
  cima--;
```

PilaArray iii

```
    }

    public boolean llena() {
      return cima == datos.length - 1;
    }

    public boolean vacia() {
      return cima == -1;
    }
  }
```

TestPilaArray ¡

```
public class TestPilaArray {
  public static void main(String[] args) {
    final int N = 10;
    PilaAcotada<String> p = new PilaArray<String>(N);
    String dato;
    assert p.vacia();
    assert !p.llena();
    for (int i = 1; i < N; i++) {
      dato = "Dato-"+i;
      p.apilar(dato);
      assert !p.vacia();
```

TestPilaArray ii

```
    assert !p.llena();
    assert p.cima().equals(dato);
  }
  dato = "Dato-"+N;
  p.apilar(dato);
  assert p.llena();
  for (int i = N; i > 1; i--) {
    dato = "Dato-"+i;
    assert !p.vacia();
    assert p.cima().equals(dato);
    p.desapilar();
    assert !p.llena();
```

TestPilaArray iii

```
        }
        dato = "Dato-"+1;
        assert !p.vacia();
        assert !p.llena();
        assert p.cima().equals(dato);
        p.desapilar();
        assert p.vacia();
        assert !p.llena();
    }
}
```

Para después de la tarea

Capturando más de una excepción

```
try {
  for (int i = 1; i < N; i++) {
    dato = "Dato-"+i;
    p.apilar(dato);
    assert !p.vacia();
    assert !p.llena();
    assert p.cima().equals(dato);
  }
}
```

Capturando más de una excepción

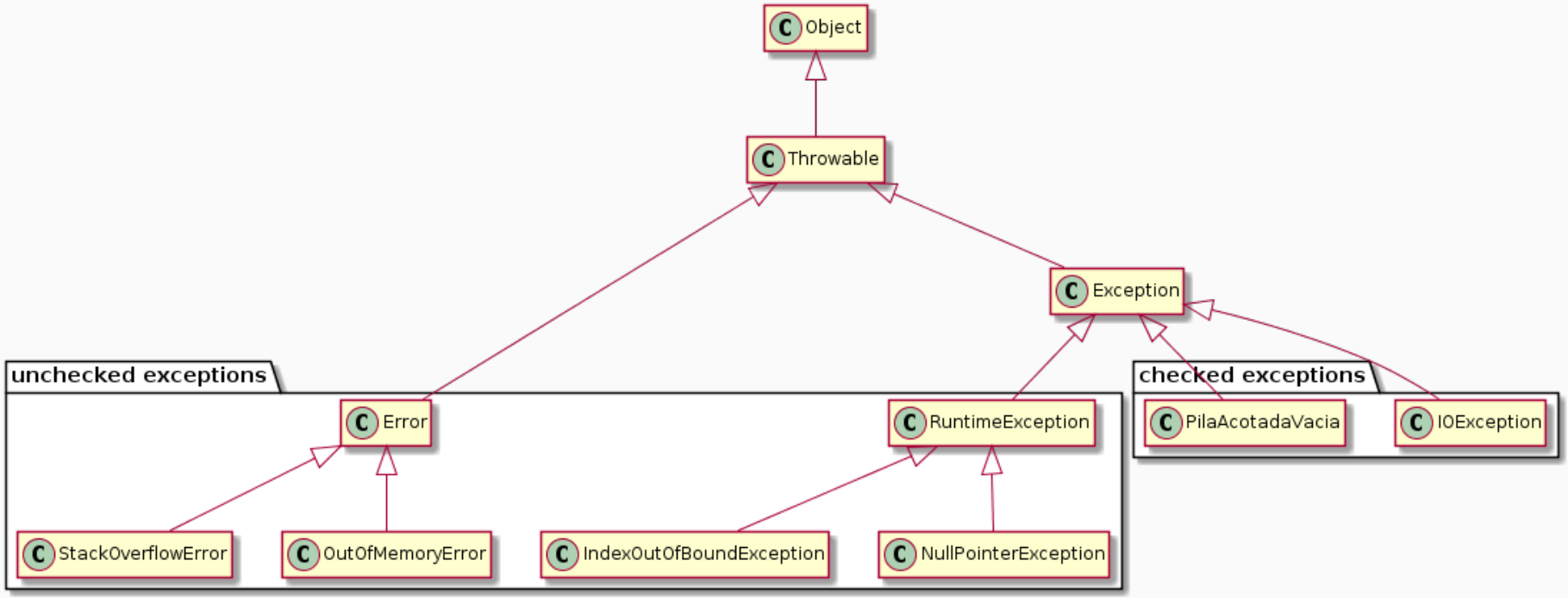
```
try {
  for (int i = 1; i < N; i++) {
    dato = "Dato-"+i;
    p.apilar(dato);
    assert !p.vacia();
    assert !p.llena();
    assert p.cima().equals(dato);
  }
}
catch (PilaAcotadaLlena excepLlena) {
  System.err.println("Error al apilar en una pila llena");
  System.exit(1);
}
```

Capturando más de una excepción

```
try {
  for (int i = 1; i < N; i++) {
    dato = "Dato-"+i;
    p.apilar(dato);
    assert !p.vacia();
    assert !p.llena();
    assert p.cima().equals(dato);
  }
}
catch (PilaAcotadaLlena excepLlena) {
  System.err.println("Error al apilar en una pila llena");
  System.exit(1);
}
catch (PilaAcotadaVacia excepVacia) {
  System.err.println("Error al sacar la cima de una pila vacía");
  System.exit(1);
}
```

Jerarquía de excepciones

- *checked*: capturar o elevar, el compilador avisa
- *unchecked*: el compilador no avisa



Aprovechando la herencia

```
try {
  for (int i = 1; i < N; i++) {
    dato = "Dato-"+i;
    p.apilar(dato);
    assert !p.vacia();
    assert !p.llena();
    assert p.cima().equals(dato);
  }
}
```

Aprovechando la herencia

```
try {
  for (int i = 1; i < N; i++) {
    dato = "Dato-"+i;
    p.apilar(dato);
    assert !p.vacia();
    assert !p.llena();
    assert p.cima().equals(dato);
  }
}
catch (Exception e) {
  // PilaAcotadaLlena y PilaAcotadaVacia heredan de Exception
  // y cualquiera de las dos se capturan en este bloque
  System.err.println("Error en el manejo de pilas");
  System.exit(1);
}
```

Excepciones con detalles del problema

```
public class PilaAcotadaLlena extends Exception
{
  private int datosEnPila;
  public PilaAcotadaLlena(int n) {
    datosEnPila = n;
  }
  public datosEnPila() {
    return datosEnPila;
  }
}
```

Algunos métodos interesantes

- `String getMessage()`
  Returns the detail message string of this throwable
- `StackTraceElement[] getStactTrace()`
  Provides programmatic access to the stack trace information printed by `printStackTrace()`
- `void printStackTrace(PrintStream s)`
  Prints this throwable and its backtrace to the standard error stream

Inspeccionando la excepción

```
try {
  for (int i = 1; i < N; i++) {
    dato = "Dato-"+i;
    p.apilar(dato);
    assert !p.vacia();
    assert !p.llena();
    assert p.cima().equals(dato);
  }
}
catch (Exception e) {
  System.err.print("Se ha detectado un error: ");
  System.err.println(e.getMessage());
  e.printStackTrace(System.err);
  System.exit(1);
}
```

Finalizando

```
try {
  for (int i = 1; i < N; i++) {
    dato = "Dato-"+i;
    p.apilar(dato);
    assert !p.vacia();
    assert !p.llena();
    assert p.cima().equals(dato);
  }
}
catch (PilaAcotadaLlena excepLlena) {
  System.err.println("Error al apilar en una pila llena");
}
catch (PilaAcotadaVacia excepVacia) {
  System.err.println("Error al sacar la cima de una pila vacía");
}
```

Finalizando

```
try {
  for (int i = 1; i < N; i++) {
    dato = "Dato-"+i;
    p.apilar(dato);
    assert !p.vacia();
    assert !p.llena();
    assert p.cima().equals(dato);
  }
}
catch (PilaAcotadaLlena excepLlena) {
  System.err.println("Error al apilar en una pila llena");
}
catch (PilaAcotadaVacia excepVacia) {
  System.err.println("Error al sacar la cima de una pila vacía");
}
finally (PilaAcotadaVacia excepVacia) {
  // El bloque "finally" se ejecuta siempre!
  System.err.println("Revisa tus tests");
  System.exit(1);
}
```