

# Sesión 22: Arrays redimensionables

## Programación 2

### 7. Implementación de TADs lineales

---

Ángel Herranz

2019-2020

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

- TADs lineales **no acotados**
- Pilas no acotadas **redimensionando** arrays
- Librería los TADs de la asignatura `https://moodle.upm.es/titulaciones/oficiales/course/view.php?id=6288&section=13`

# Pilas acotadas

- API: crear, apilar, desapilar, cima, está vacía y está llena
- Pila: interfaz genérico
- PilaVacíaException: excepción para cima y desapilar
- ~~PilaAcotadaLlena~~: excepción para apilar
- PilaRedimensionable: implementación de Pila
- TestPila: tests para Pila

 Redimensionar el array cuando sea necesario

# Código de apoyo

---

# Pila

```
public interface Pila<T> {  
    void apilar(T dato);  
    T cima() throws PilaVacíaException;  
    void desapilar() throws PilaVacíaException;  
    boolean vacía();  
}
```

# PilaVacíaException

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

# TestPila: crear y llenando

```
public class TestPila {  
    public static void main(String[] args) {  
        final int N = 100_000;  
        Pila<String> p = new PilaRedimensionable<String>();  
        String dato;  
        assert p.vacia();  
        for (int i = 1; i < N; i++) {  
            dato = "Dato-"+i;  
            p.apilar(dato);  
            assert !p.vacia();  
            try { assert p.cima().equals(dato); }  
            catch (PilaVacíaException e) { assert false; }  
        }  
        dato = "Dato-"+N;  
        p.apilar(dato);  
    }  
}
```



# TestPila: casi vaciando

```
for (int i = N; i > 1; i--) {  
    dato = "Dato-"+i;  
    assert !p.vacia();  
    try {  
        assert p.cima().equals(dato);  
        p.desapilar();  
    }  
    catch (PilaVacíaException e) {  
        assert false;  
    }  
}  
assert !p.vacia();
```

# TestPila: vaciando

```
dato = "Dato-"+1;
try {
    assert p.cima().equals(dato);
    p.desapilar();
}
catch (PilaVacíaException e) {
    assert false;
}
assert p.vacia();
```

# TestPila: excepciones

```
try {  
    p.cima();  
    assert false;  
}  
catch (PilaVacíaException e) { }
```

```
try {  
    p.desapilar();  
    assert false;  
}  
catch (PilaVacíaException e) { }  
}  
}
```

# PilaRedimensionable: de uno en uno

- Un único atributo: **private** T[] datos
- El array se crea con longitud 0
- Se redimensiona en 1 al apilar
- Se redimensiona en 1 al desapilar
- La cima es la posición `datos.length-1`

# PilaRedimensionable: de uno en uno

- Un único atributo: **private** T[] datos
- El array se crea con longitud 0
- Se redimensiona en 1 al apilar
- Se redimensiona en 1 al desapilar
- La cima es la posición `datos.length-1`

 20 minutos

# PilaRedimensionable: de uno en uno

```
angel@T440p: /Sesión20 $ javac TestPila.java
angel@T440p: /Sesión20 $ time -p java -ea TestPila
real 12.79
user 12.40
sys 0.43
angel@T440p: /Sesión20 $ cowsay -p "Okay, Houston, we've had a problem here"

-----
< Okay, Houston, we've had a problem here >
-----
      \  ^__^
       \  (oo)\_______
          (__)\       )\/\
              ||----w |
              ||     ||

angel@T440p: /Sesión20 $
```


## PilaRedimensionable: de $N$ en $N$

- Otro atributo: **int** cima
- El array se crea con longitud  $N$
- Se redimensiona en  $N$  al apilar
- Se redimensiona en  $N$  al desapilar cuando se desaprovecha espacio

## PilaRedimensionable: de 1, 2, 4, 8, 16, ...

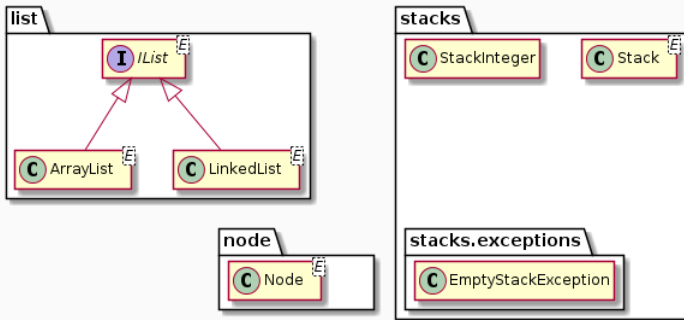
- En vez de redimensionar linealmente ...
- se puede redimensionar **exponencialmente**

 ¿Para qué sirve esta aproximación?

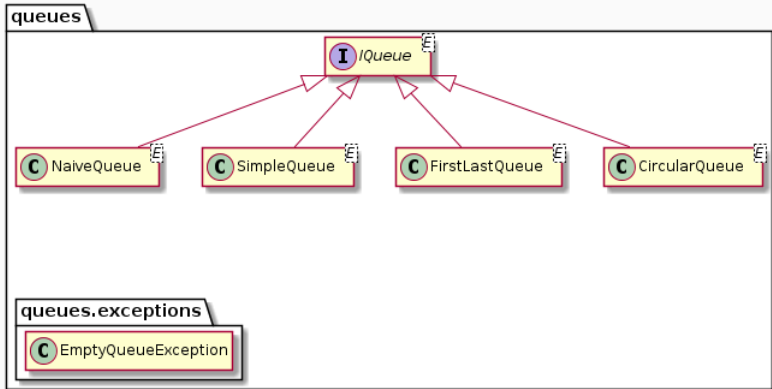
 ¿Qué problemas tiene?



# En capítulos posteriores: `tads1920.jar` i



# En capítulos posteriores: `tads1920.jar` ii





# Explorar `tads1920.jar`

- Biblioteca y **javadocumentación**:

<https://moodle.upm.es/titulaciones/oficiales/mod/folder/view.php?id=764645>

- Implementación **para practicar**:

<https://moodle.upm.es/titulaciones/oficiales/course/view.php?id=6288&section=13>

- Ejemplo:

