SEMINARIO

5

Ejercicios

Diseño del Software

Grado en Ingeniería Informática del Software

Curso 2017-2018

Escuela de Ingeniería Informática
Universidad de Oviedo

César Acebal

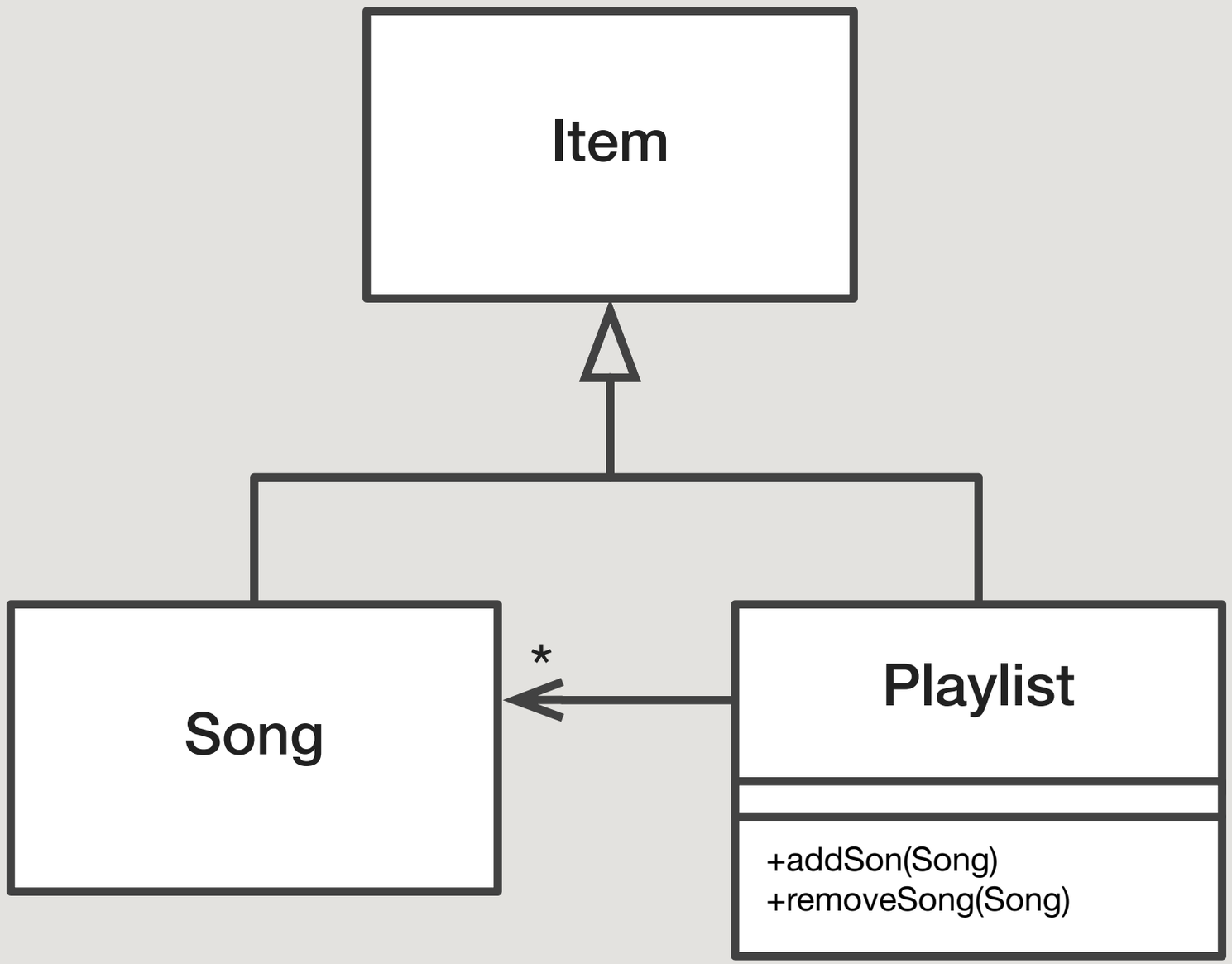
Canciones, carpetas y listas de reproducción

¿Cómo sería el diseño? ¿Qué patrón es?

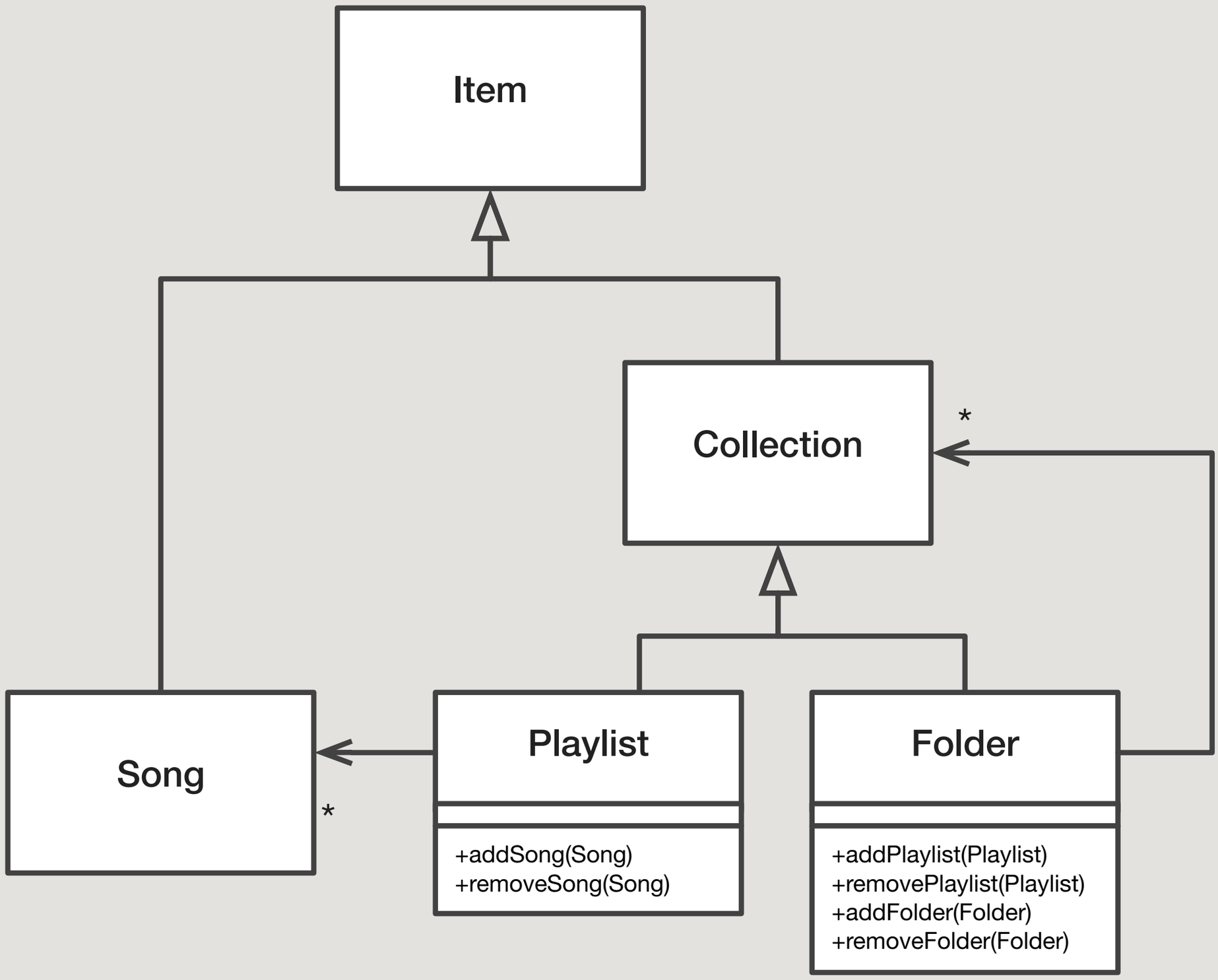
Canciones

- **Estamos haciendo un organizador y reproductor multimedia, tipo iTunes, que puede tener:**
  - Canciones
  - Listas de reproducción
  - Carpetas
    - Que pueden contener otras carpetas y listas de reproducción (pero no canciones sueltas)
- **¿Cómo sería?**

Sólo con canciones y listas



Con las carpetas



¿Qué patrón de diseño es?

Canciones, vídeos y sus reproductores asociados

Canciones

- **Ahora suponed que añadimos también la posibilidad de albergar y reproducir vídeos a nuestro reproductor multimedia:**
  - Canciones
  - Vídeos

Reproductores

- Las canciones y los vídeos tienen cada uno su tipo de reproductor asociado
- ¿Cómo mejoraríamos el siguiente diseño?

```
public class ListaDeReproduccion
{
  ...
  public void reproducir()
  {
    Iterator<Elemento> iterador = elementos.iterator();
    while (iterator.hasNext()) {
      Elemento elemento = iterator.next();
      Reproductor reproductor;
      if (elemento instanceof Video)
        reproductor = new ReproductorDeVideo();
      else if (elemento instanceof Cancion)
        reproductor = new ReproductorDeAudio();
      reproductor.reproducir(elemento);
  }
  ...
}
```

¿Qué patrón de diseño es?

Iteradores en la API de Java

El método «iterator» de las colecciones de Java...
¿qué patrón de diseño es?

Filtros en Java I/O

Veamos cómo se podría añadir una nueva clase a la biblioteca de E/S de Java que convierta todo lo que se escribe en ella en minúsculas

Java I/O

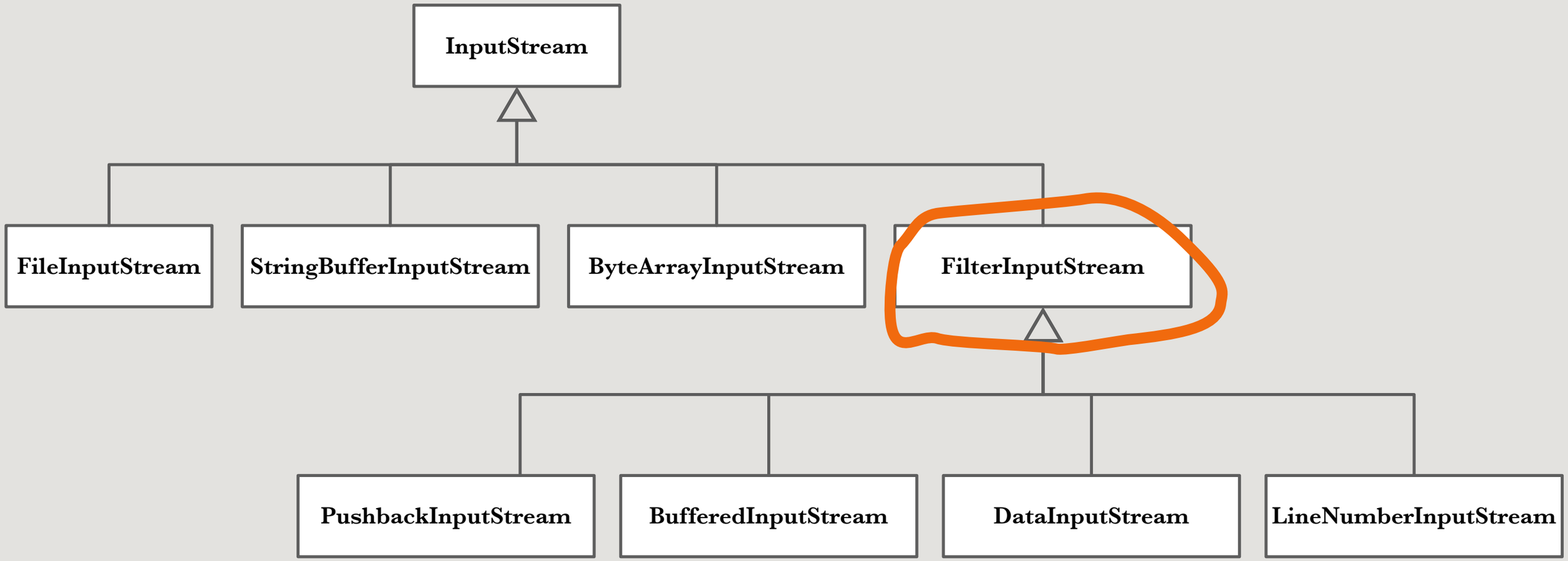
```
Reader in = new BufferedReader
    (new InputStreamReader(System.in));
```

- **Ejemplos:**
  - BufferedInputStream
  - LineNumberInputStream
  - DataInputStream
  - PushbackInputStream

¿Qué tienen en común estas clases?

Sí, son decoradores concretos.
Pero, ¿qué clase de la API de Java haría las veces del decorador en sí?

Decoradores en Java I/O



Escribir nuestro propio decorador

Escribir un decorador que funcione con el resto de clases de la biblioteca de entrada/salida de Java y que convierta a minúsculas todo lo que lea.

- **Pistas:**
  - Heredar de la clase **`FilterInputStream`**
  - Implementar los dos métodos **`read`**
- **Lo probáis con un pequeño método main que por ejemplo lea un fichero de texto** (usando vuestro decorador, se entiende)
  - Y vais escribiendo cada carácter leído en la consola

Solución

```
import java.io.*;

public class LowerCaseInputStream extends FilterInputStream {

    public LowerCaseInputStream(InputStream in) {
        super(in);
    }

    public int read() throws IOException {
        int c = super.read();
        return (c == -1 ? c : Character.toLowerCase((char)c));
    }

    public int read(byte[] b, int offset, int len) throws IOException {
        int result = super.read(b, offset, len);
        for (int i = offset; i < offset+result; i++) {
            b[i] = (byte)Character.toLowerCase((char)b[i]);
        }
        return result;
    }
}
```

```
import java.io.*;

public class InputTest {
    public static void main(String[] args) throws IOException {
        int c;

        try {
            InputStream in =
                new LowerCaseInputStream(
                    new BufferedInputStream(
                        new FileInputStream("test.txt")));

            while((c = in.read()) >= 0) {
                System.out.print((char)c);
            }

            in.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Framework de pruebas

Ejercicio

- Un framework de pruebas implementa en su clase `TestCase` un método `test`
- Los programadores deben heredar de esa clase para implementar sus propias pruebas unitarias
- Los creadores del framework quieren que ese método siempre ejecute un código de inicialización y de liberación de recursos antes y después del código de prueba en sí

¿Cómo sería?

```
public abstract class TestCase
{
    ...

    public void run()
    {
        setUp();
        runTest();
        tearDown();
    }
```

```
    protected abstract void
      runTest();

    protected void setUp()
    {
    }

    protected void tearDown()
    {
    }

    ...
}
```

¿Qué patrón de diseño es?