POLITÉCNICA

"Ingeniamos el futuro"

CAMPUS
DE EXCELENCIA
INTERNACIONAL

MiW

Patrones de Diseño

28. Ejercicios

Luis Fernández Muñoz

https://www.linkedin.com/in/luisfernandezmunyoz

setillofm@gmail.com

Códigos

Todos los códigos ejemplo de los patrones se encuentran en:

https://github.com/miw-upm/IWVG/tree/master/doo/src/main/java/designPatterns

Todos los códigos de ejercicios sobre patrones se encuentran en:

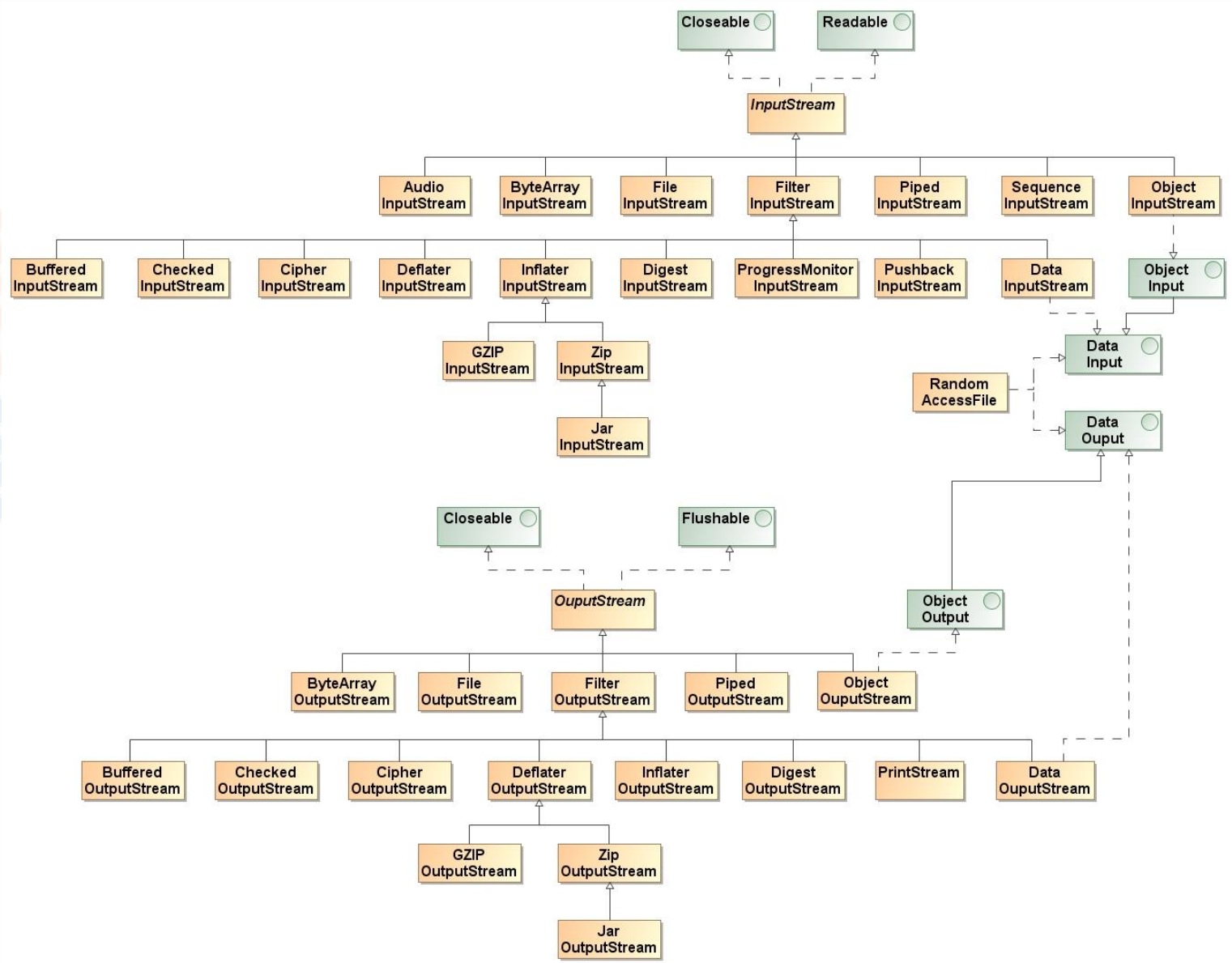
https://github.com/miw-upm/IWVG/tree/master/doo/src/main/java/desingPatterns/exercises

INDICE

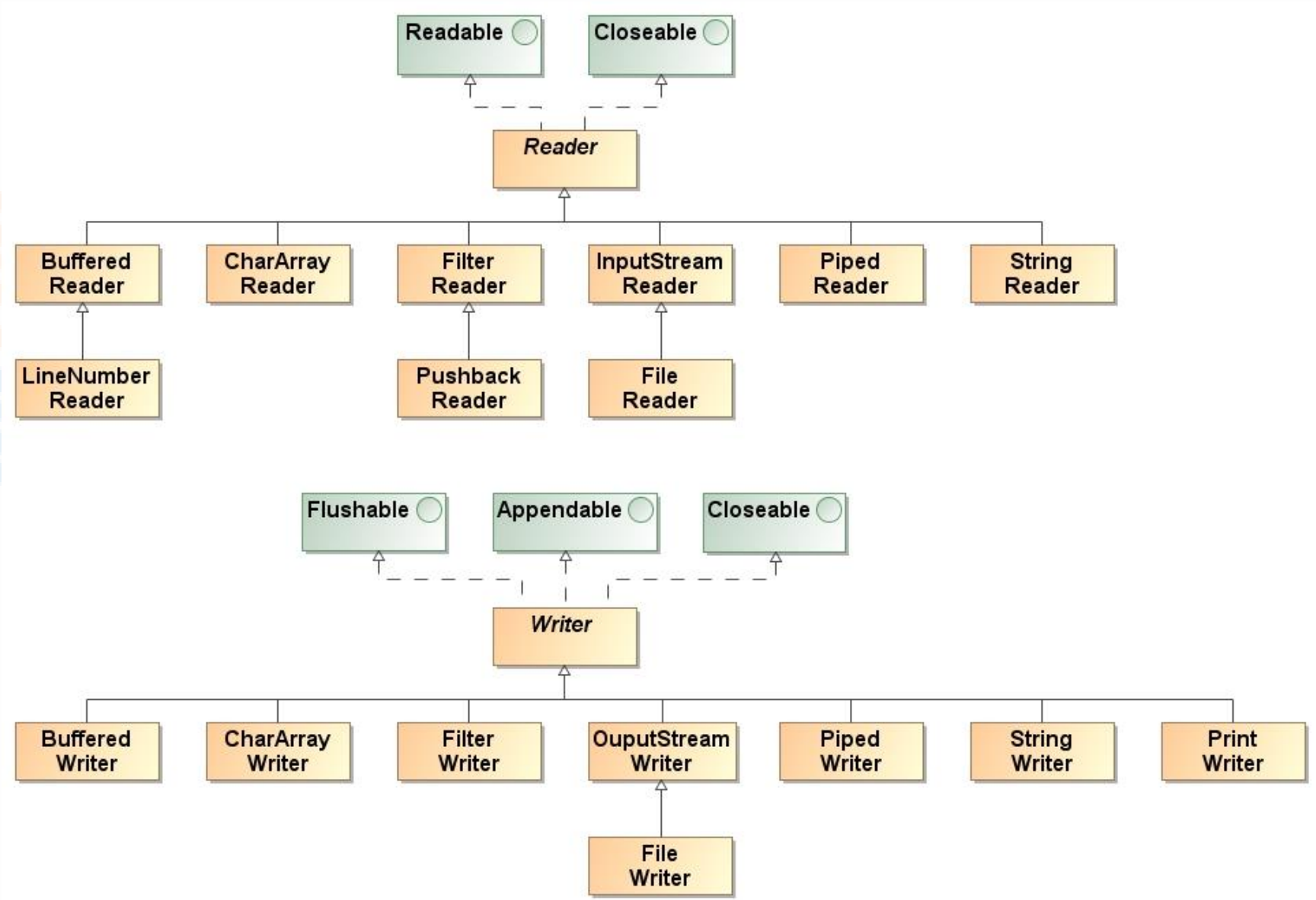
1. Uso del Subsistema java.io

- Los flujos (*streams*) son objetos que representan secuencias ordenadas de datos que tienen una fuente (flujos de entrada) o un destino (flujos de salida);
- Las clases de entrada/salida del paquete java.io se definen en términos de flujos, y permiten a los programadores abstraerse de los detalles específicos del sistema operativo al acceder a los recursos del sistema, tales como ficheros, pantalla, teclado, líneas de comunicaciones, etc.;
- En la biblioteca se distinguen dos grandes tipos de flujos:
  - de entrada y salida de bytes, cuyas clases abstractas de las que heredan todas las demás son *InputStream* y *OutputStream*;
  - de entrada y salida de caracteres1, cuyas clases abstractas de las que heredan todas las demás son *Reader* y *Writer*;

1. Uso del Subsistema java.io



1. Uso del Subsistema java.io



1. Uso del Subsistema java.io

- **Enunciado**: Sus posibilidades son inmensas pero en la mayoría de las aplicaciones se requiere un uso muy particular respecto de todas las posibilidades.
  - Este uso se obtiene con la colaboración de objetos de varias clases
  - Esto complica y repite código en todos aquellos clientes que requieran las funcionalidades del subsistema

1. Uso del Subsistema java.io

- **Considerar cómo los patrones de diseño que resuelven problemas**
  - Determinando la granularidad de los objetos: ***Facade***
- **Analizar las secciones de intención**
  - *Facade:* Proveer de una interfaz unificada a un conjunto de interfaces de un subsistema definiendo una interfaz de más alto nivel que hace más fácil de usar el subsistema
- **Estudiar cómo los patrones se interrelacionan**
  - *Singleton*
- **Estudiar el propósito de los patrones**
  - ~~*Facade vs Mediator*~~
  - ~~*Facade vs Abstract Factory*~~
- **Examina las causas de rediseño**
  - Fuerte Acoplamiento

1. Uso del Subsistema java.io

- **Considerar que sería variable en el diseño**
  - La interfaz de un subsistema
- **Leer el patrón como una visión general**
- **Estudiar las secciones de Estructura, Participantes y Colaboradores**
- **Leer la sección de Código de Ejemplo para ver un ejemplo concreto del patrón en código**
- **Elegir nombre para los participantes del patrón que sean significativos en el contexto de la aplicación**
  - IO.class
- **Definir las clases**

1. Uso del Subsistema java.io

- **Definir nombres específicos de la aplicación para las operaciones en el patrón**
  - void write(<*tipoPrimitivo*>);
  - void writeln(<*tipoPrimitivo*>);
  - <*tipoPrimitivo*> *read*<*TipoPrimitivo*>();
- **Implementar las operaciones que lleven a cabo las responsabilidades y colaboraciones en el patrón**