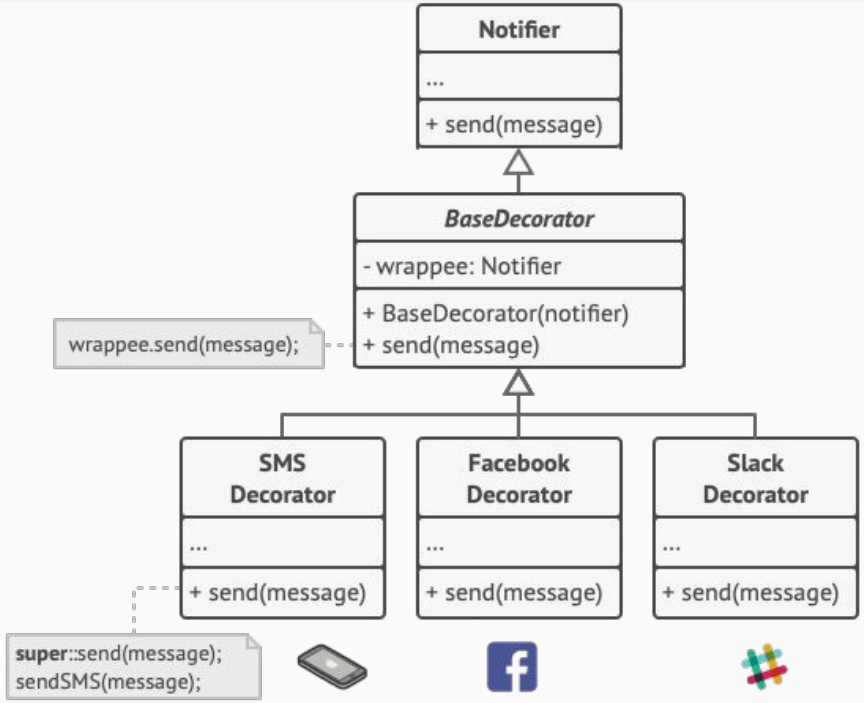
Java

Работа с файлами и потоками. Часть 2.

Декоратор


Notifier
...
+ send(message)
BaseDecorator
- wrappee: Notifier
+ BaseDecorator(notifier)
+ send(message)
wrappee.send(message);
SMS Decorator
...
+ send(message)
Facebook Decorator
...
+ send(message)
Slack Decorator
...
+ send(message)
super::send(message);
sendSMS(message);

Декоратор

- Принимает в конструкторе интерфейс
- Реализует тот же интерфейс/базовый класс
- Вызывает тот же метод у обёрнутого класса

Шаблон Builder

- Используется для объектов с большим количеством параметров
- Может использоваться вместо большого конструктора
- Может использоваться для предобработки параметров
- Может использоваться для создания неизменяемых объектов

Потоки данных

- InputStream
- OutputStream

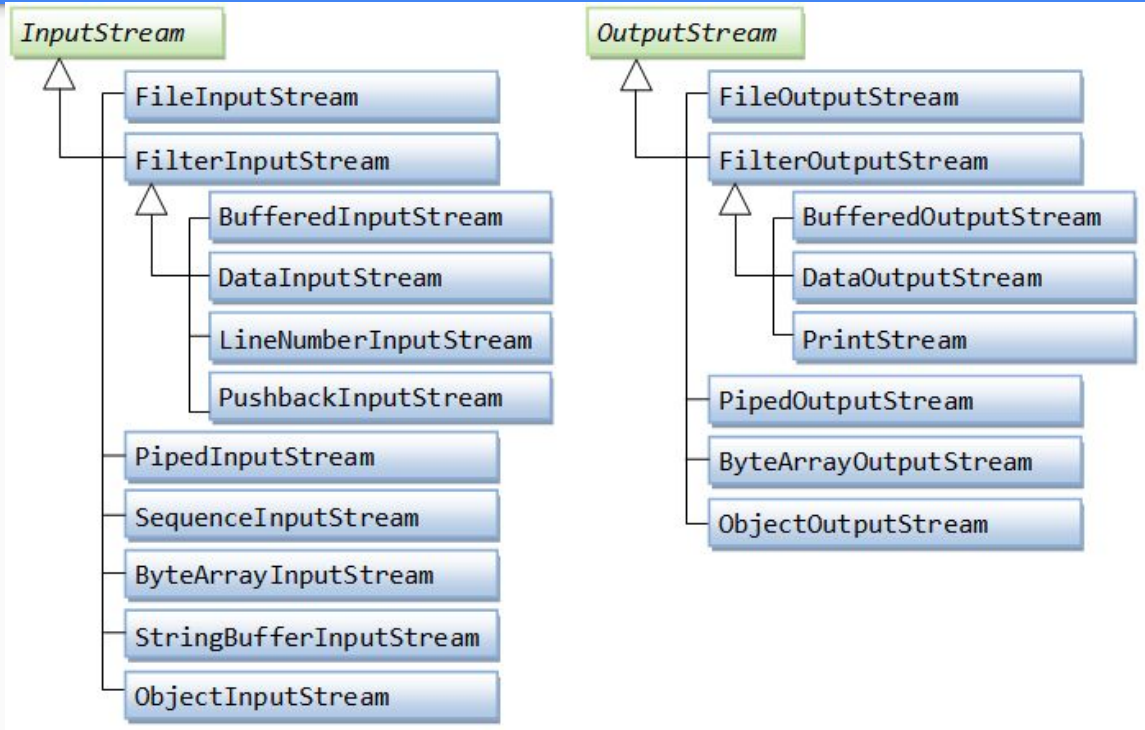
InputStream

- read()
- read(byte[] data)
- read(byte[] data, int offset, int length)
- close()

OutputStream

- write(int b)
- write(byte[] data)
- write(byte[] data, int offset, int length)
- flush()
- close()

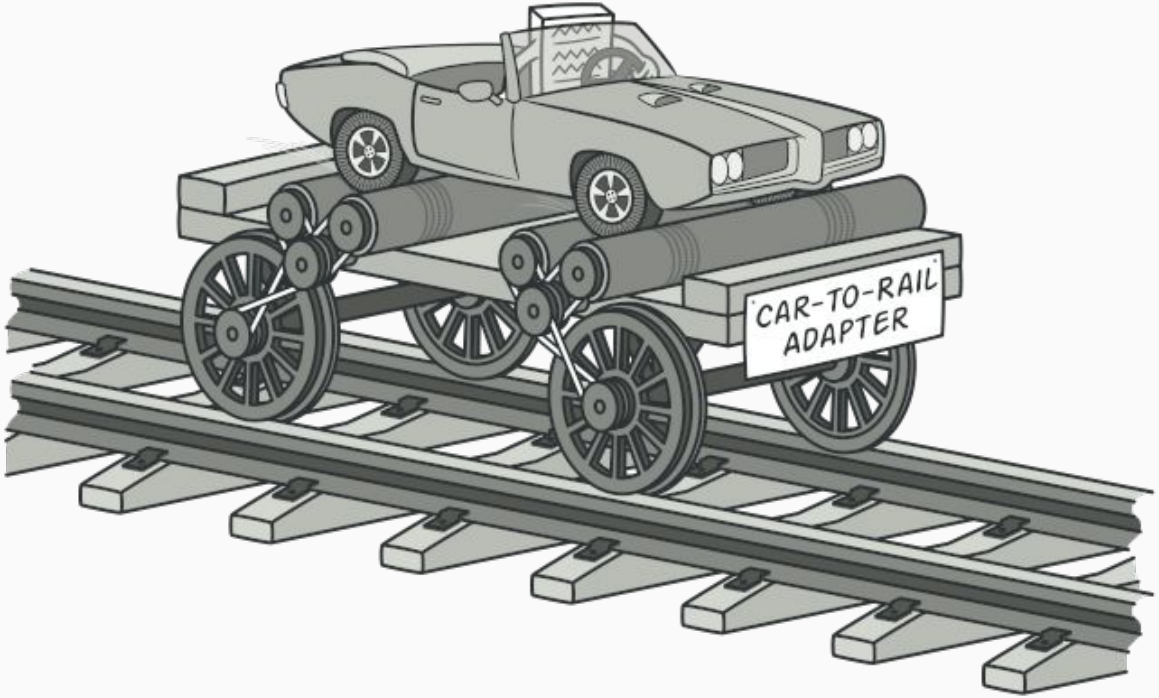
Потоки данных



Потоки символов

- *Reader
- *Writer

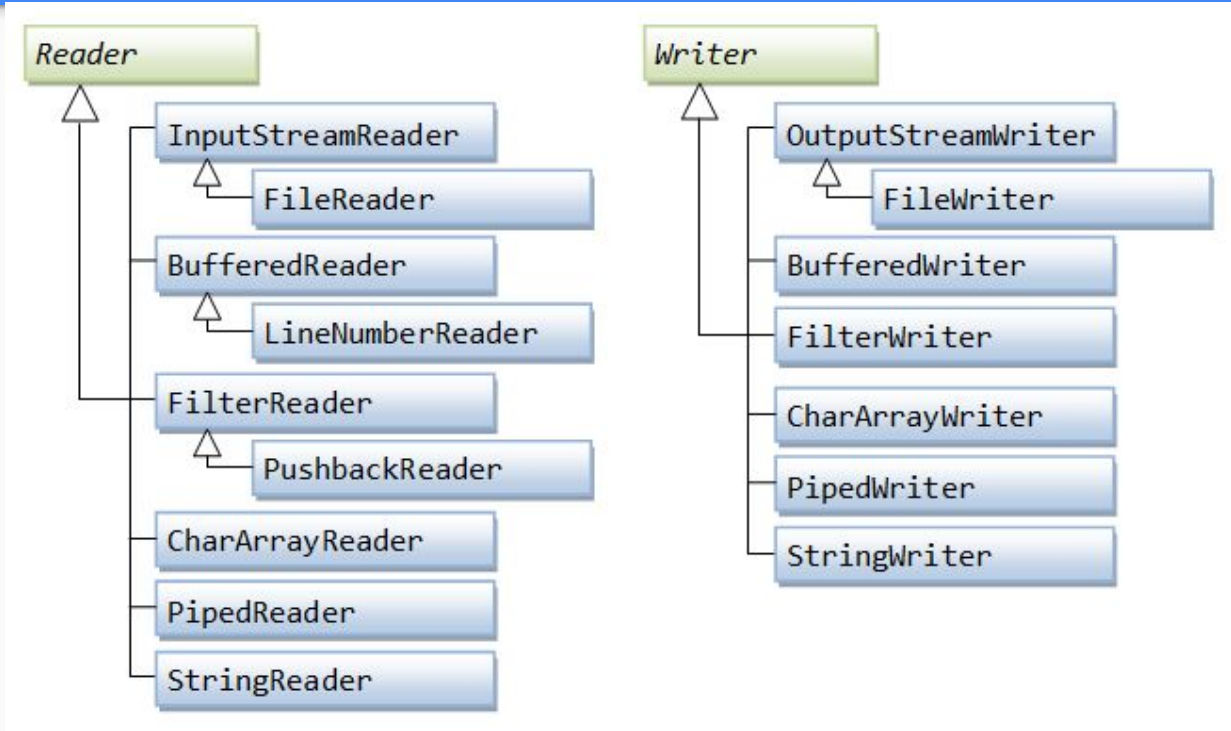
Шаблон адаптер



Адаптер

- Позволяет совместить один интерфейс с другим
- Используется когда один сервис работает с одним типом, а данные хранятся в объекте другого типа
- Пример из Java - InputStreamReader

Потоки символов



PrintWriter

- print(Object)
- println(Object)
- printf(String, Object…)
- Как вывод в консоль, но может использовать любой поток

System

- InputStream in - поток ввода
- PrintStream out - поток вывода
- PrintStream err - поток ошибок (пишется красным)

Потоки символов

- Reader reader = new StringReader(“Hello World!”);
- StreamTokenizer tokenizer = new StreamTokenizer(reader);
- Считается устаревшим, рекомендуется пользоваться методом split()

Scanner

Scanner

```
Scanner scanner = new Scanner (System.in);

scanner.nextInt();
```

- Может использоваться в том числе для обработки файла