Java

Занятие 7. File, IO/NIO, Stream

File

- Находится в java.io
- File file = new File("filePath");

File

- file.getName();
- file.isFile();
- file.isDirectory();
- file.exists();

File

```
if (file.isDirectory()){
    File[] files = file.listFiles();
}
```

File

```
file.mkdir();
file.mkdirs();

file.delete();
```

File

. - текущая директория

.. - родительская директория

File

- a\b\..\file.txt
- a\..\b\c\file.txt
- .\a\b\..\b\c\.\file.txt
- a\.\b\..\c\.\file.txt
- a\b\c\file.txt

File: как сравнить

- file.getCanonicalPath();

NIO

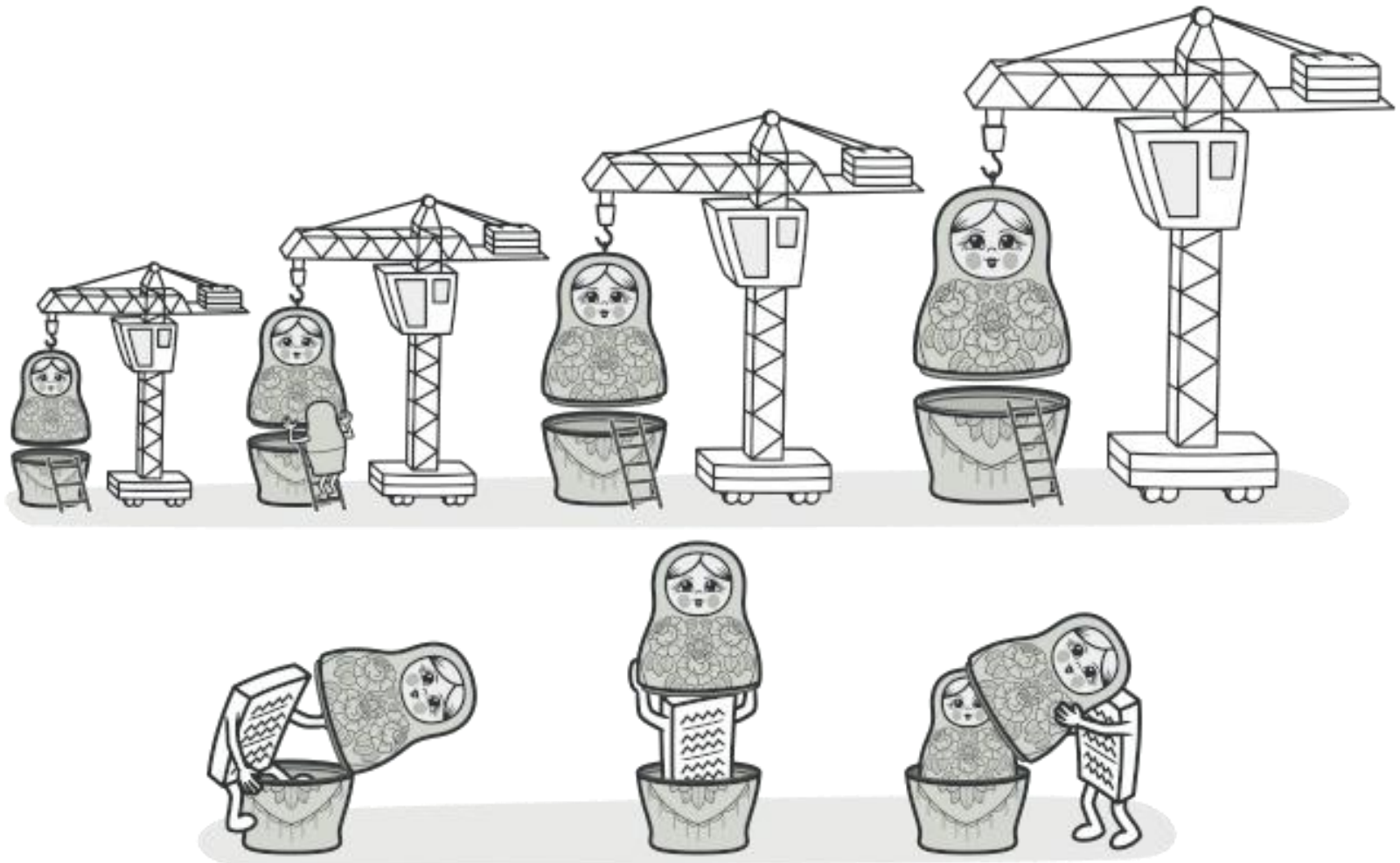
- Paths
- Files

NIO

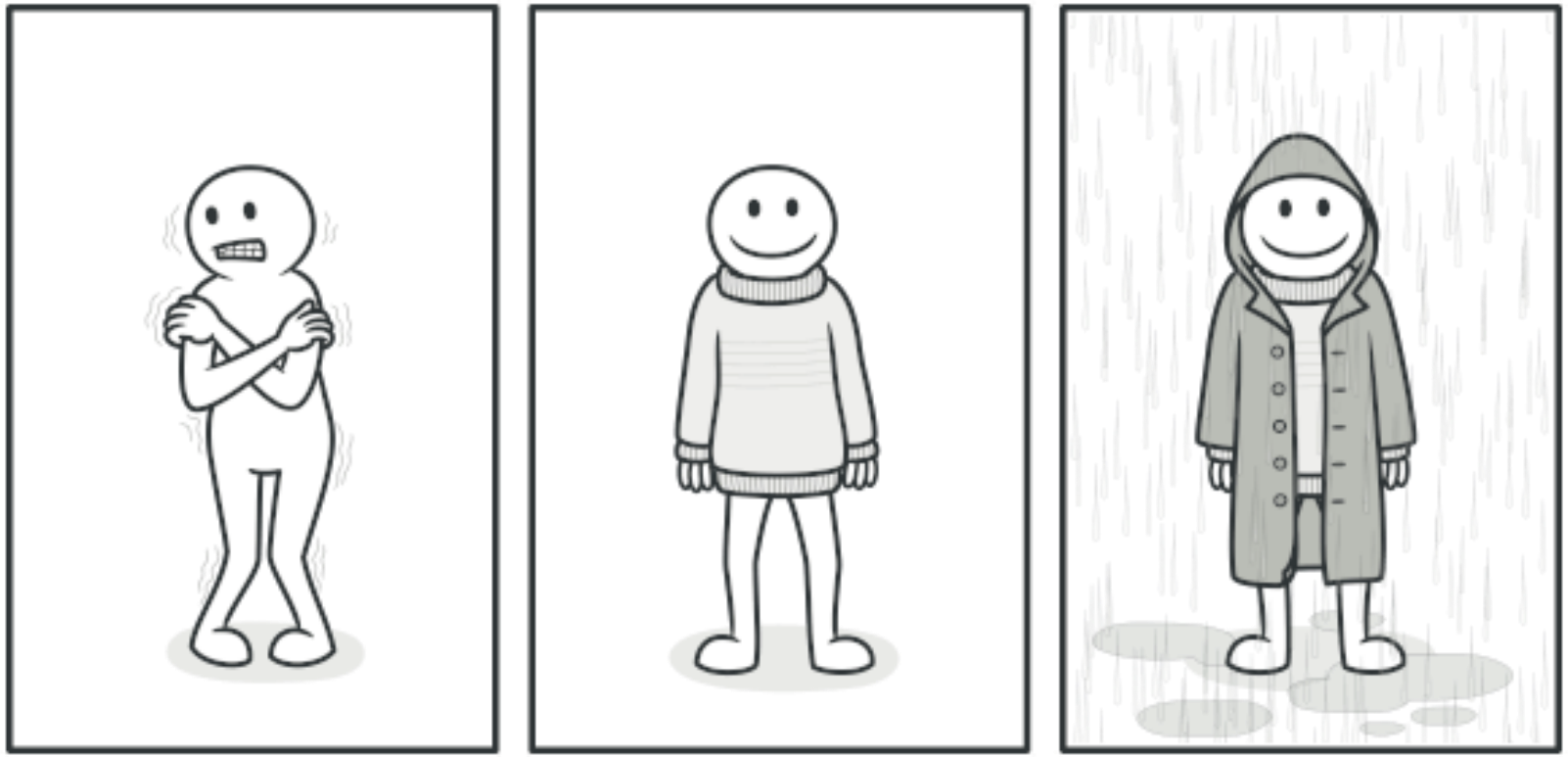
```
Path path = Paths.get("./target/");
Files.walkFileTree(path, new SimpleFileVisitor<Path>() {
    @Override
    public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) throws
IOException {
        Files.delete(file);
        return FileVisitResult.CONTINUE;
    }

    @Override
    public FileVisitResult postVisitDirectory(Path dir, IOException exc)
            throws IOException {
        Files.delete(dir);
        return FileVisitResult.CONTINUE;
    }
});
```

Декоратор



Декоратор



Потоки данных

- InputStream
- OutputStream

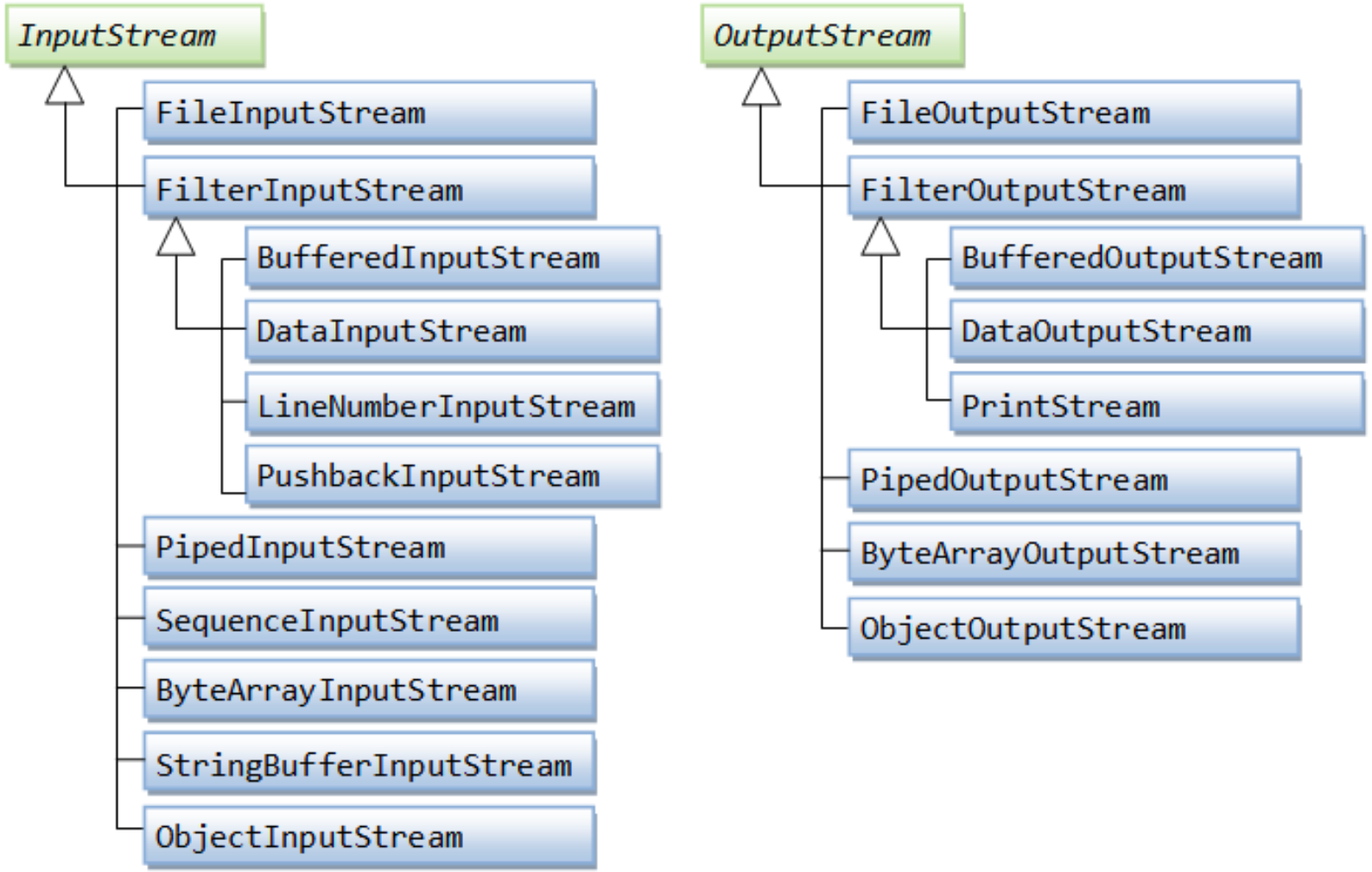
InputStream

- **public abstract int** read() **throws** IOException;
- **public int** read(**byte**[] var1) **throws** IOException;
- **public int** read(**byte**[] var1, **int** var2, **int** var3) **throws** IOException;
- **public void** close() **throws** IOException;

OutputStream

- **public abstract int** write() **throws** IOException;
- **public int** write(**byte**[] var1) **throws** IOException;
- **public int** write(**byte**[] var1, **int** var2, **int** var3) **throws** IOException;
- **public void** flush() **throws** IOException;
- **public void** close() **throws** IOException;

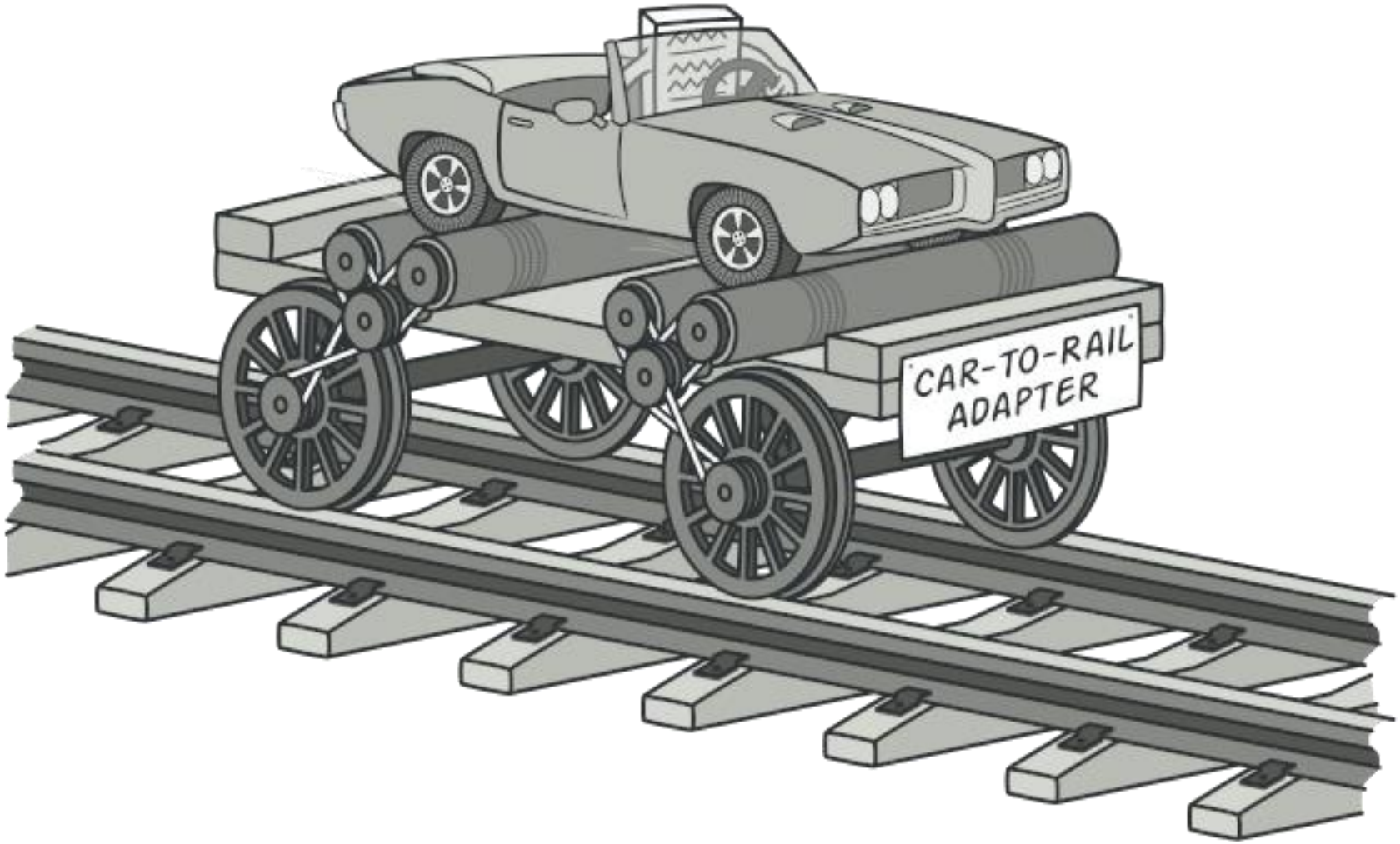
Потоки данных



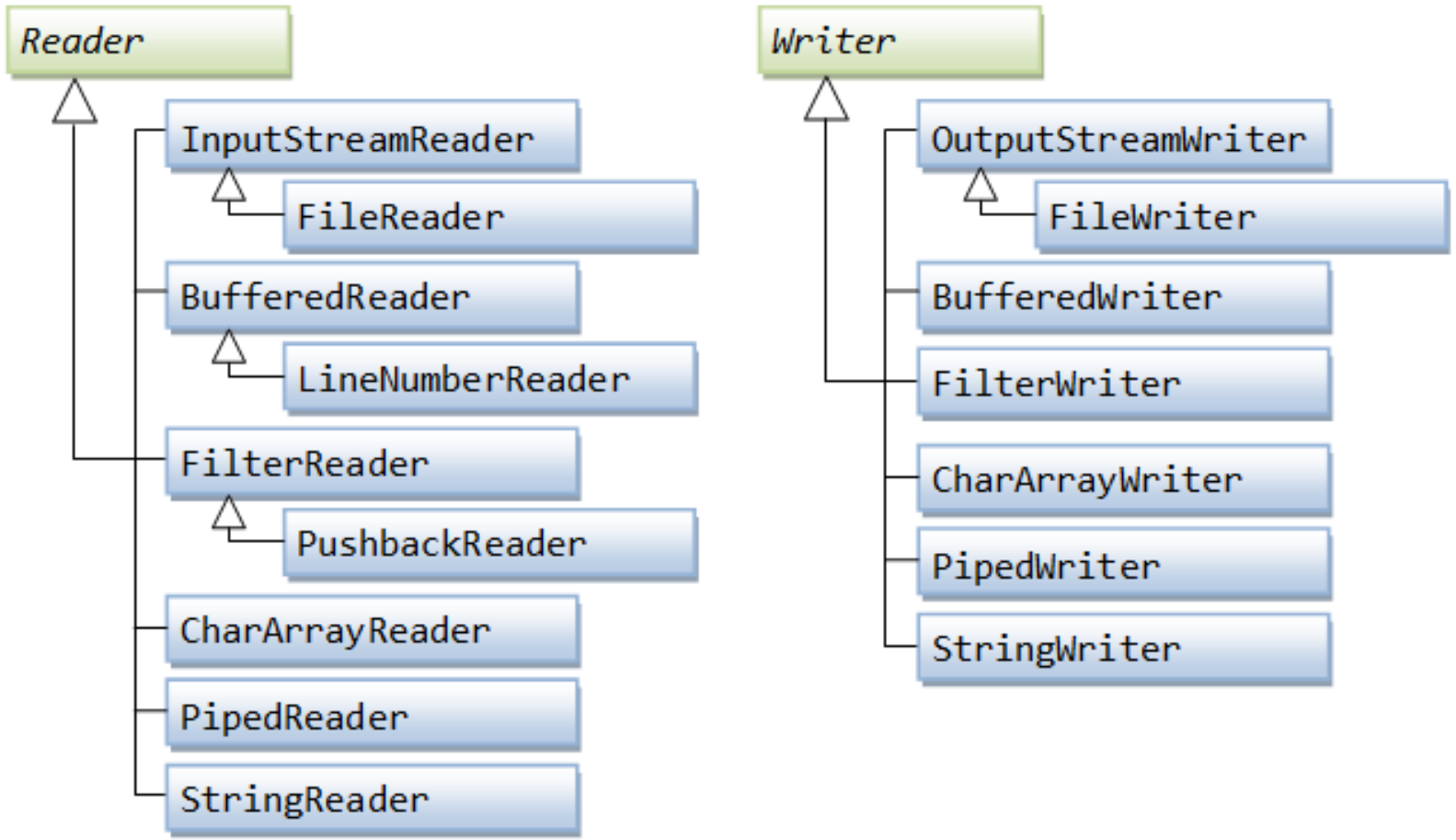
Потоки символов

- Reader
- Writer

Адаптер



Потоки символов



PrintWriter

- **public void** print(Object var1);
- **public void** println(Object var1);
- **public** PrintWriter printf(String var1, Object... var2);

Потоки символов

```
Reader reader = new StringReader("Example");
StreamTokenizer tokenizer = new StreamTokenizer(reader);
```

Scanner

Scanner

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

scanner.nextInt();
```

System

public static final InputStream in;

public static final PrintStream out;

public static final PrintStream err;

Практика

Максимальный байт:

Ввести с консоли имя файла. Найти максимальный байт в файле, вывести его на экран. Закрыть поток ввода-вывода.

Самые редкие байты:

Ввести с консоли имя файла. Найти байт или байты с минимальным количеством повторов. Вывести их на экран через пробел. Закрыть поток ввода-вывода.

Разделение файла:

Считать с консоли три имени файла: файл1, файл2, файл3. Разделить файл1 по следующему критерию: Первую половину байт записать в файл2, вторую половину байт записать в файл3. Если в файл1 количество байт нечетное, то файл2 должен содержать бОльшую часть. Закрыть потоки.

Практика

Wrapper (Decorator):

Используя шаблон
проектирования Wrapper (Decorator) расширить функциональность File OutputStream.
В классе MyFileOutputStream при вызове метода close() должна быть реализована следующая функциональность:
1. Вывести в консоль фразу "*Вы действительно хотите закрыть поток? Д/Н*".
2. Считайте строку.
3. Если считанная строка равна "*Д*", то закрыть поток.
4. Если считанная строка не равна "*Д*", то не закрывать поток.