НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Объектно-ориентированное программирование на языке Java

I / O в Java

Yevhen Berkunskyi, NUoS
Smykodub Tatiana, NUoS
eugeny.berkunsky@gmail.com
http://www.berkut.mk.ua

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Input & Output

- Создание хорошей системы ввода / вывода (I / O) является одной из наиболее сложных задач для разработчика языка.
- Об этом свидетельствует ряд различных подходов.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Класс File (java.io)

- Класс File имеет обманчивое имя; вы можете подумать, что это относится к файлу, но это не так.
- Фактически, «FilePath» было бы лучшим именем для класса.
- Он может представлять либо имя определенного файла, либо имена набора файлов в каталоге. Если это набор файлов, вы можете запросить этот набор, используя метод list (), который возвращает массив String.
- Этот класс не содержит методы для работы с содержимым файла, но позволяет манипулировать такими свойствами файла, как право доступа, дата и время создания, путь в иерархии каталогов, создание, удаление файла, изменение его имени и каталога и т. д.

Пакет java.nio.file

- Пакет java.nio.file определяет интерфейсы и классы виртуальной машины Java для доступа к файлам, атрибутам файлов и файловым системам.
- Этот API может использоваться для преодоления многих ограничений класса java.io.File. Метод toPath может использоваться для получения Path, который использует абстрактный путь, представленный объектом File, чтобы найти файл.
- Результирующий Path может использоваться с классом Files для обеспечения более эффективного и расширенного доступа к дополнительным файлам, атрибутам файлов и исключениям ввода-вывода для диагностики ошибок при сбое операции в файле.

Класс File (java.io)

Объект класса **File** создается одним из нижеприведенных способов:

```
File obFile = new File("\\com\\file.txt");
File obDir = new File("c:/jdk/src/java/io");
File obFile1 = new File(obDir, "File.java");
File obFile2 = new File("c:\\com", "file.txt");
File obFile3 = new File(new URI("Интернет-адрес"));
```

- В первом случае создается объект, соответствующий файлу, во втором — подкаталогу. Третий и четвертый случаи идентичны.
- Для создания объекта указывается каталог и имя файла. В пятом — создается объект, соответствующий адресу в Интернете.

При создании объекта класса **File** любым из конструкторов компилятор не выполняет проверку на существование физического файла с заданным путем.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Пример

```
package by.bsu.io;
import java.io.File;
import java.util.Date;
import java.io.IOException;
public class FileTest {
public static void main(String[] args) {
// с объектом типа File ассоциируется файл на диске
FileTest2.java
File fp = new File("FileTest2.java");
if(fp.exists()) {
System.out.println(fp.getName() + " существует");
if(fp.isFile()) { // если объект – дисковый файл
System.out.println("Путь к файлу:\t" + fp.getPath());
System.out.println("Абсолютный путь:\t" + fp.getAbsolutePath());
System.out.println("Размер файла:\t" + fp.length());
System.out.println("Последняя модификация :\t"+ new
Date(fp.lastModified()));
System.out.println("Файл доступен для чтения:\t" + fp.canRead());
System.out.println("Файл доступен для записи:\t" +
fp.canWrite());
System.out.println("Файл удален:\t" + fp.delete());
}}
```

Пример

```
else
System.out.println("файл " + fp.getName() + " не существует");
try {
if(fp.createNewFile()) {
System.out.println("Файл " + fp.getName() + " создан");
}
} catch(IOException e) {
System.err.println(e);
}
// в объект типа File помещается каталог\директория
// в корне проекта должен быть создан каталог com.learn с
несколькими файлами
File dir = new File("com" + File.separator + "learn");
if (dir.exists() && dir.isDirectory()) { /*если объект
является каталогом и если этот
каталог существует */
System.out.println("каталог " + dir.getName() + "
существует");
}
```

Пример

```
File[] files = dir.listFiles();
for(int i = 0; i < files.length; i++) {
Date date = new Date(files[i].lastModified());
System.out.print("\n"+files[i].getPath()+" \t|
"+files[i].length()+"\t|"+date);
// использовать toLocaleString() или toGMTString()
}
// метод listRoots() возвращает доступные корневые
каталоги
File root = File.listRoots()[1];
System.out.printf("\n%s %,d из %,d свободно.",
root.getPath(), root.getUsableSpace(),
root.getTotalSpace());
}
}
```

Input и output

- Классы библиотеки Java ввода-вывода делятся на ввод и вывод, согласно иерархии классов в документации JDK.
- Через наследование все производные от классов **InputStream** или **Reader** имеют базовые методы, называемые **read** () для чтения одного байта или массива байтов.
- Аналогично, все производные от классов **OutputStream** или **Writer** имеют базовые методы, называемые **write** () для записи одного байта или массива байтов.

InputStream & OutputStream

- В java.io разработчики библиотеки начали с решения, что все классы, имеющие какое-либо отношение к вводу, будут унаследованы от **InputStream**, а все классы, связанные с выводом, будут унаследованы от **OutputStream**.

Типы InputStream

В классе **InputStream** определены свойства, общие для байтовых потоков ввода.

Базовый класс **InputStream** представляет классы, которые получают данные из различных источников:

- массив байтов
- строка (String)
- файл
- канал (pipe): данные помещаются с одного конца и извлекаются с другого
- последовательность различных потоков, которые можно объединить в одном потоке
- другие источники (например, подключение к интернету)

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Обзорная таблица классов Java IO

	С битами		С символами	
	Ввод	Вывод	Ввод	Вывод
Базовые	InputStream	OutputStream	Reader InputStreamReader	Writer OutputStreamWriter
Массивы	ByteArrayInputStream	ByteArrayOutputStream	CharArrayReader	CharArrayWriter
Файлы	FileInputStream RandomAccessFile	FileOutputStream RandomAccessFile	FileReader	FileWriter
Каналы	PipedInputStream	PipedOutputStream	PipedReader	PipedWriter
Буфер	BufferedInputStream	BufferedOutputStream	BufferedReader	BufferedWriter
Фильтрация	FilterInputStream	FilterOutputStream	FilterReader	FilterWriter
Синтакс. анализ Parsing	PushbackInputStream StreamTokenizer		PushbackReader LineNumberReader	
Строки			StringReader	StringWriter
Работ а с примитивами	DataInputStream	DataOutputStream		
		PrintStream		PrintWriter
Работа с сериал. объектами	ObjectInputStream	ObjectOutputStream		
Объединение InputStream	SequenceInputStream			

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

InputStream

Для работы с указанными источниками используются подклассы базового класса **InputStream**:

- `BufferedInputStream` - Буферизированный входной поток
- `ByteArrayInputStream` – Позволяет использовать буфер в памяти (массив байтов) в качестве источника данных для входного потока.
- `DataInputStream` - Входной поток, включающий методы для чтения стандартных типов данных
- `FileInputStream` - Для чтения информации из файла
- `FilterInputStream` - Абстрактный класс, предоставляющий интерфейс для классов-надстроек, которые добавляют к существующим потокам полезные свойства.
- `InputStream` - Абстрактный класс, описывающий поток ввода
- `ObjectInputStream` - Входной поток для объектов
- `StringBufferInputStream` - Превращает строку (String) во входной поток данных InputStream

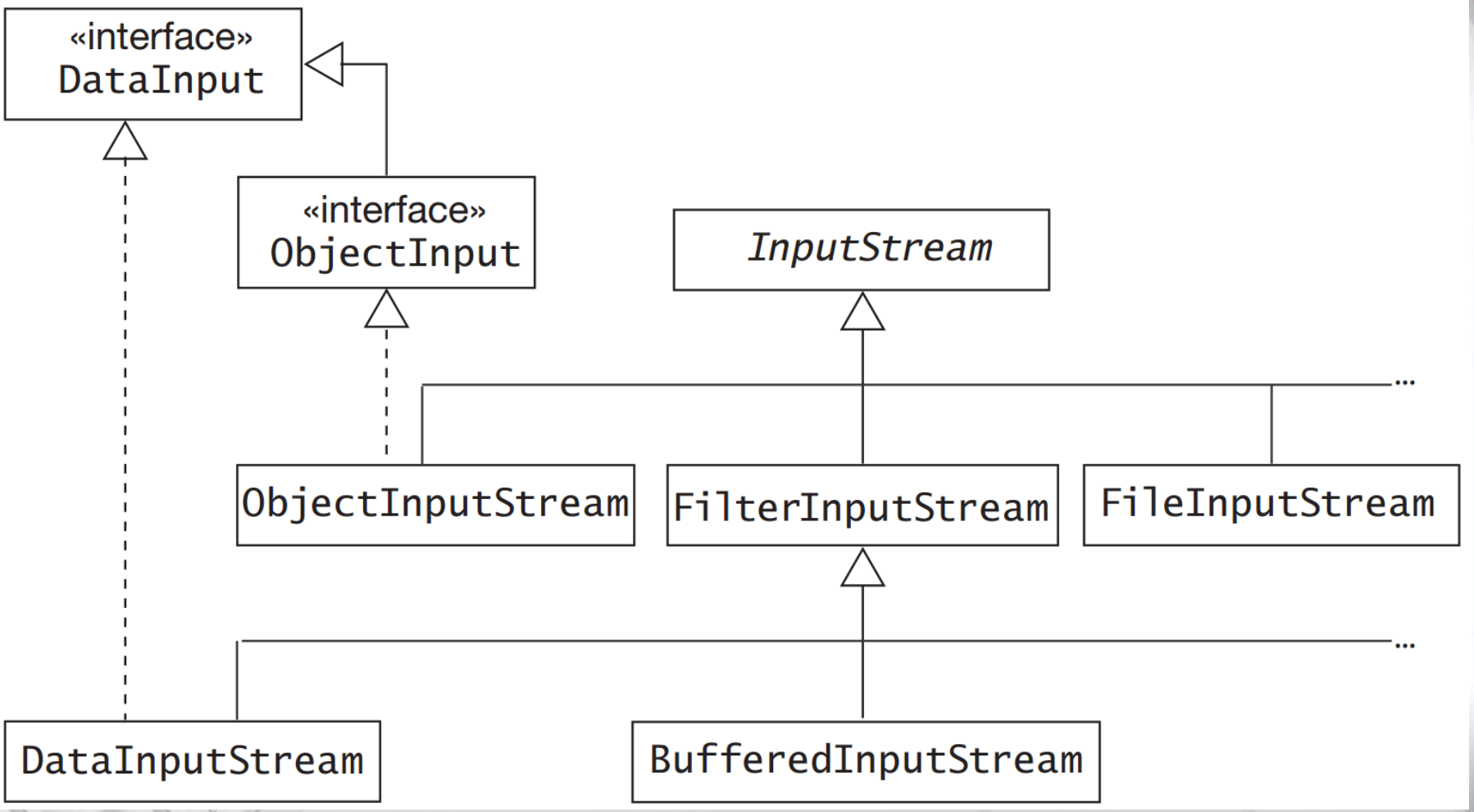
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

OutputStream

Класс **OutputStream** - это абстрактный класс, определяющий потоковый байтовый вывод. В этой категории находятся классы, определяющие, куда направляются ваши данные: в массив байтов (но не напрямую в String; предполагается что вы сможете создать их из массива байтов), в файл или канал.

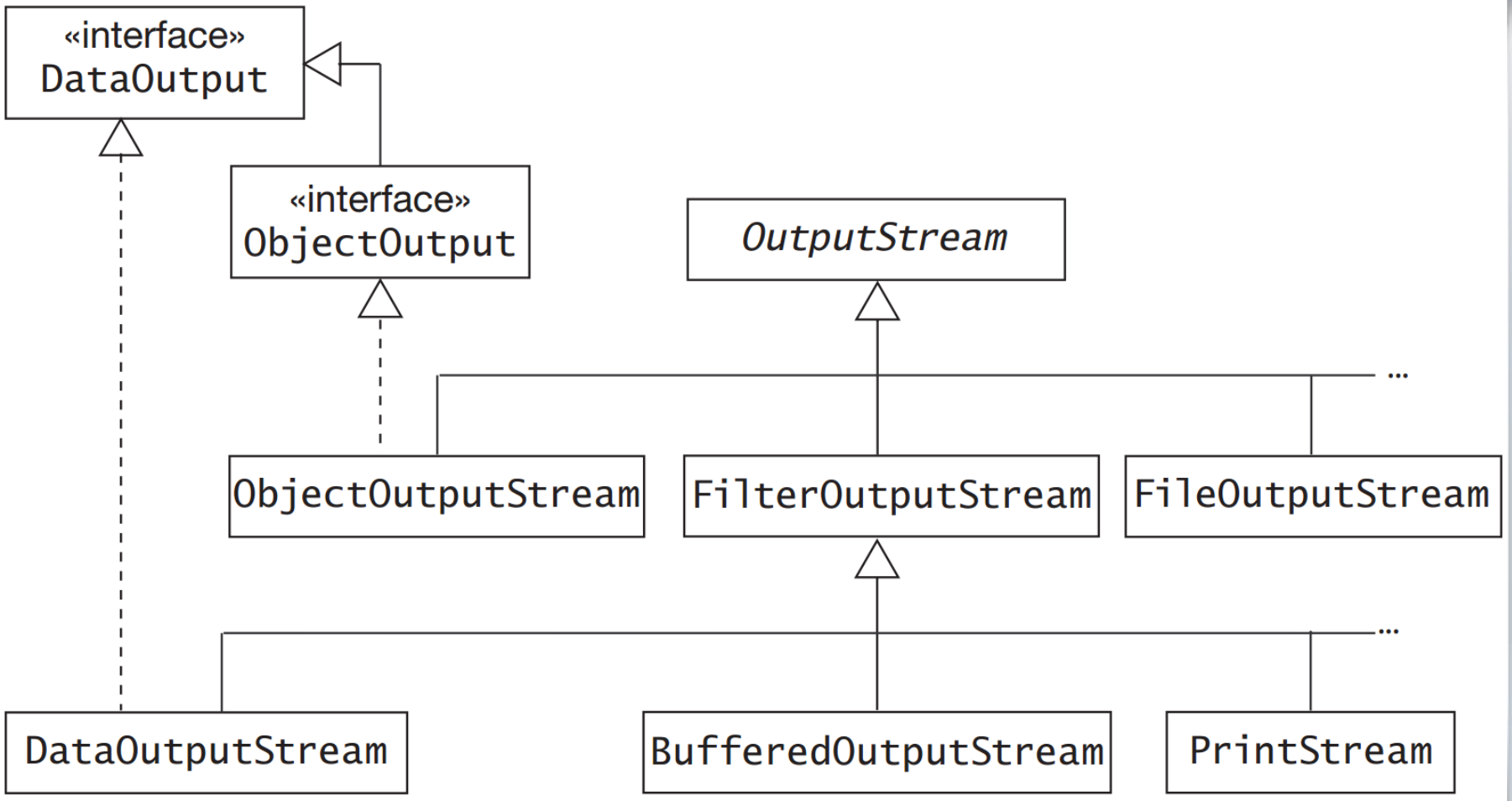
- `BufferedOutputStream` - Буферизированный выходной поток
- `ByteArrayOutputStream` - Создает буфер в памяти. Все данные, посылаемые в этот поток, размещаются в созданном буфере
- `DataOutputStream` - Выходной поток, включающий методы для записи стандартных типов данных
- `JavaFileOutputStream` - Отправка данных в файл на диске. Реализация класса OutputStream
- `ObjectOutputStream` - Выходной поток для объектов
- `PipedOutputStream` - Реализует понятие выходного канала.
- `FilterOutputStream` - Абстрактный класс, предоставляющий интерфейс для классов-надстроек, которые добавляют к существующим потокам полезные свойства.

Иерархия InputStreams



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Иерархия OutputStreams



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Символьные потоки: чтение и запись

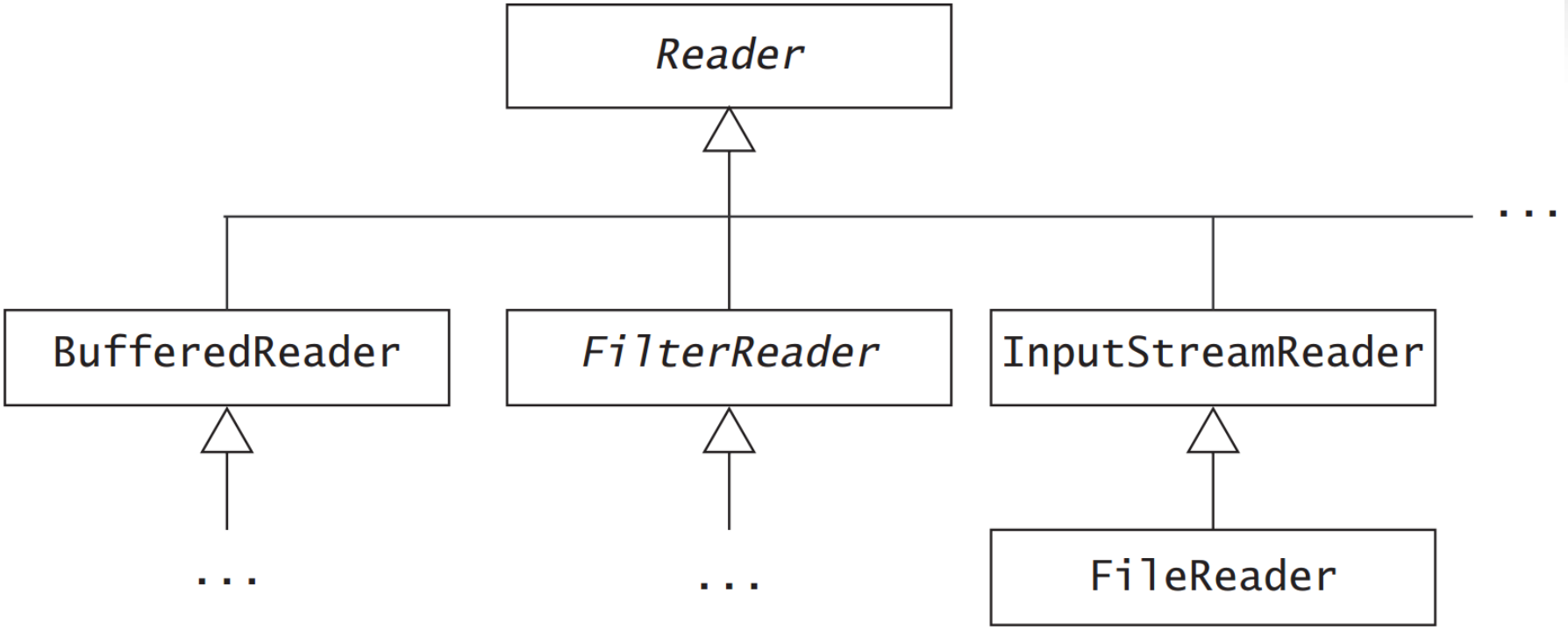
- Причиной появления классов **Reader** и **Writer** стала интернационализация, старая библиотека поддерживала 8 битное представление символов и неправильно иногда работала с 16 - ти. Для правильной обработки символьных потоков в формате Unicode применяется отдельная иерархия подклассов абстрактных классов **Reader** и **Writer**, которые почти полностью повторяют функциональность байтовых потоков, но являются более актуальными при передаче текстовой информации.
- **Reader** представляет собой поток входных символов, который считывает последовательность символов Unicode, а **Writer** - это выходной поток символов, который записывает последовательность символов Unicode .

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Чтение

BufferedReader	Читатель, который буферизует символы, читаемые от основного читателя. Необходимо указать базовый читатель и указать необязательный размер буфера.
InputStreamReader	Символы считываются из байтового входного потока, который должен быть указан. Кодировка символов по умолчанию используется, если явно не указано кодирование символов.
FileReader	Читает символы из файла, используя кодировку по умолчанию. Файл может быть задан объектом File, FileDescriptor или именем файла String. Он автоматически создает FileInputStream, связанный с файлом.

Character Streams: Readers and Writers



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Запись

BufferedWriter	Писатель, который буферизирует символы, прежде чем записывать их в основной писатель. Основной писатель должен быть указан, и может быть указан необязательный размер буфера.
OutputStreamWriter	Символы записываются в поток байтов, который должен быть указан. Используется кодировка символов по умолчанию, если не указано явное кодирование символов.
FileWriter	Записывает символы в файл, используя кодировку по умолчанию. Файл может быть задан объектом File, FileDescriptor или именем файла String. Он автоматически создает FileOutputStream, связанный с файлом
PrintWriter	Фильтр, позволяющий записывать текстовое представление объектов Java и примитивных типов Java в основной поток. Необходимо указать основной выходной поток или писатель.

Character Streams: Readers and Writers