УНИВЕРСИТЕТ ИТМО

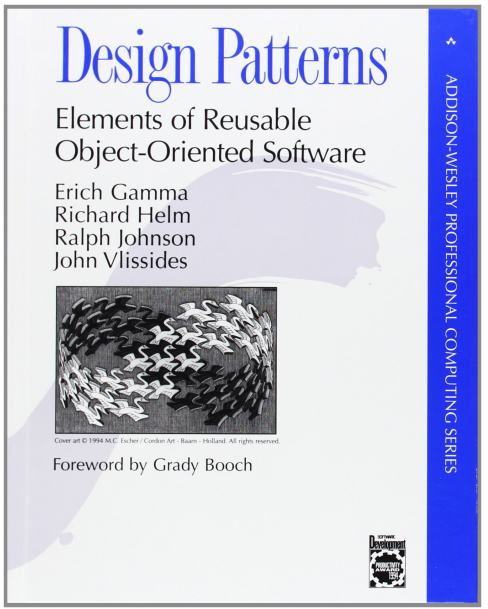
Программирование. 2 семестр

Шаблоны проектирования (введение)

ITMO more than a UNIVERSITY

Шаблоны проектирования

- ☑ GoF Book (Gang of Four)
- ☑ Банда четырех



- ☑ Шаблон — эффективное решение задачи в определенном контексте
  - Контекст — ситуация
  - Задача — цель + ограничения
  - Решение — что сделать
- ☑ Антишаблон — неэффективное решение часто встречающихся задач

Java

Фундаментальные шаблоны

- ☑ Делегирование (Delegation)
- ☑ Интерфейс-метка (Marker Interface)
- ☑ Неизменяемый объект (Immutable)
- ☑ Интерфейс + абстрактный суперкласс (Interface + Abstract Superclass)

Делегирование и наследование

- ☑ Наследование - подкласс другого класса
  - собака - это подкласс животного
  - отношение **is-a**

```
class Dog extends Animal  { }
```

Делегирование и наследование

- ☑ Наследование - подкласс другого класса
  - собака - это подкласс животного
  - отношение **is-a**
- ☑ Типы Student и Person
  - студент - не подкласс, а **роль**

```
class Dog extends Animal  { }

class Person { }

class Student extends Person { }
class Teacher extends Person { }
```

Java

Делегирование и наследование

- ☑ Наследование - подкласс другого класса
  - собака - это подкласс животного
  - отношение **is-a**
- ☑ Типы Student и Person
  - студент - не подкласс, а **роль**
  - роль может меняться
- ☑ Делегирование!

```
class Dog extends Animal  { }

class Person {
  public getName() { }
}

class Student {
  Person p;
  public getName() {
    return p.getName();
  }
}
```

Делегирование и наследование

- ☑ Наследование - подкласс другого класса
  - собака - это подкласс животного
  - отношение **is-a**
- ☑ Композиция
  - "часть-целое"
  - отношение **has-a**

```
class Dog extends Animal  { }

class Car {
  Engine engine;
  Wheel[] wheels;
  Door[] doors;
}
```

Java

Шаблон Интерфейс-метка

- ☑ Интерфейс без методов
- ☑ Помечает класс, как способный что-то делать

Java

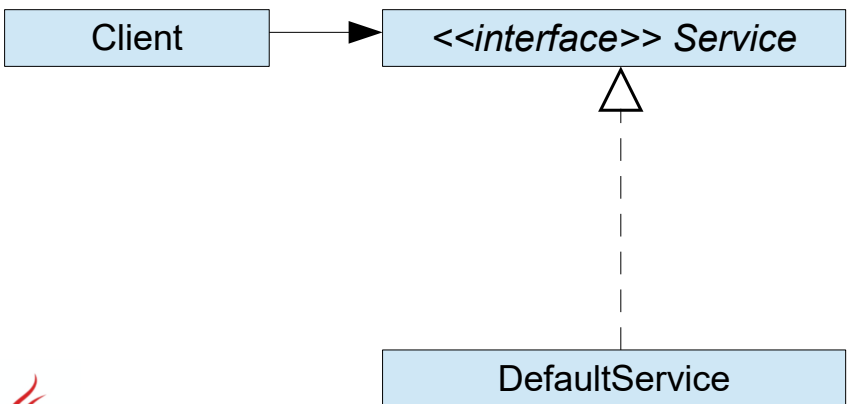
Интерфейс Cloneable

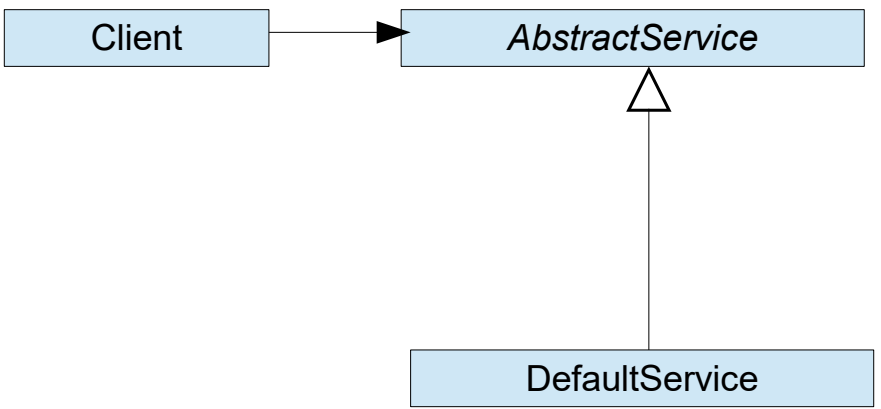
- ☑ Показывает, что разрешено вызывать clone()
- ☑ В интерфейсе Cloneable - нет метода clone()
- ☑ Object не реализует Cloneable, есть метод Object.clone()
- ☑ Object.clone() throws CloneNotSupportedException
- ☑ Неглубокое копирование - shallow copy

Java

Интерфейс / Абстрактный суперкласс

- ☑ Доступ через интерфейс исключает зависимость от реализации
- ☑ Наследование от абстрактного класса унифицирует поведение потомков и упрощает реализацию

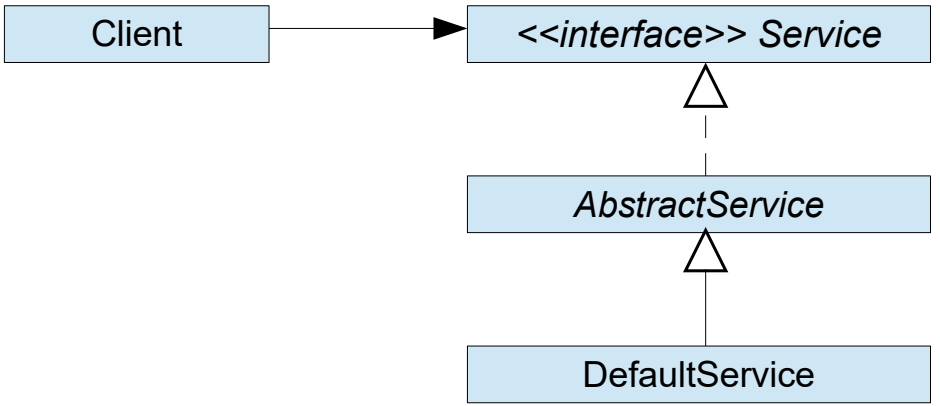




Java

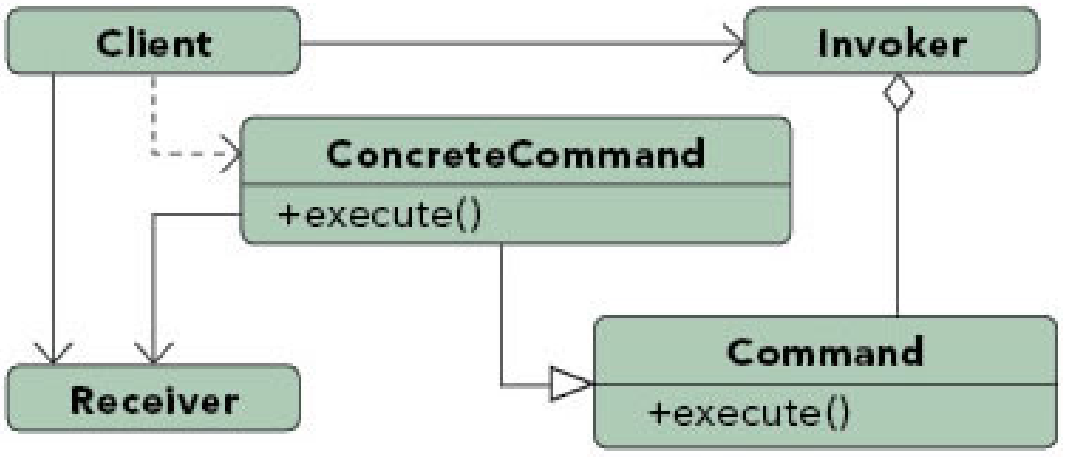
Интерфейс + Абстрактный суперкласс

- ☑ Совмещенный паттерн
- ☑ Клиент работает на уровне интерфейса
- ☑ Абстрактный класс задает базовое поведение
- ☑ Set<T> ⇠ AbstractSet<T> ⇦ HashSet<T>



Шаблон Команда (Command)

- ☑ Управление командами
- ☑ Разделение вызова и исполнения команд
- ☑ Можно организовать очередь команд и макрокоманды



УНИВЕРСИТЕТ ИТМО

Программирование. 2 семестр

Ввод-вывод

ITMO more than a UNIVERSITY

Ввод-вывод в Java

- ☑ java.io
  - Абстракция потока ввода-вывода
  - Данные - поток байтов/символов
- ☑ java.nio
  - Абстракция канала и буфера
  - Буфер - хранение данных
  - Канал - соединение для передачи данных

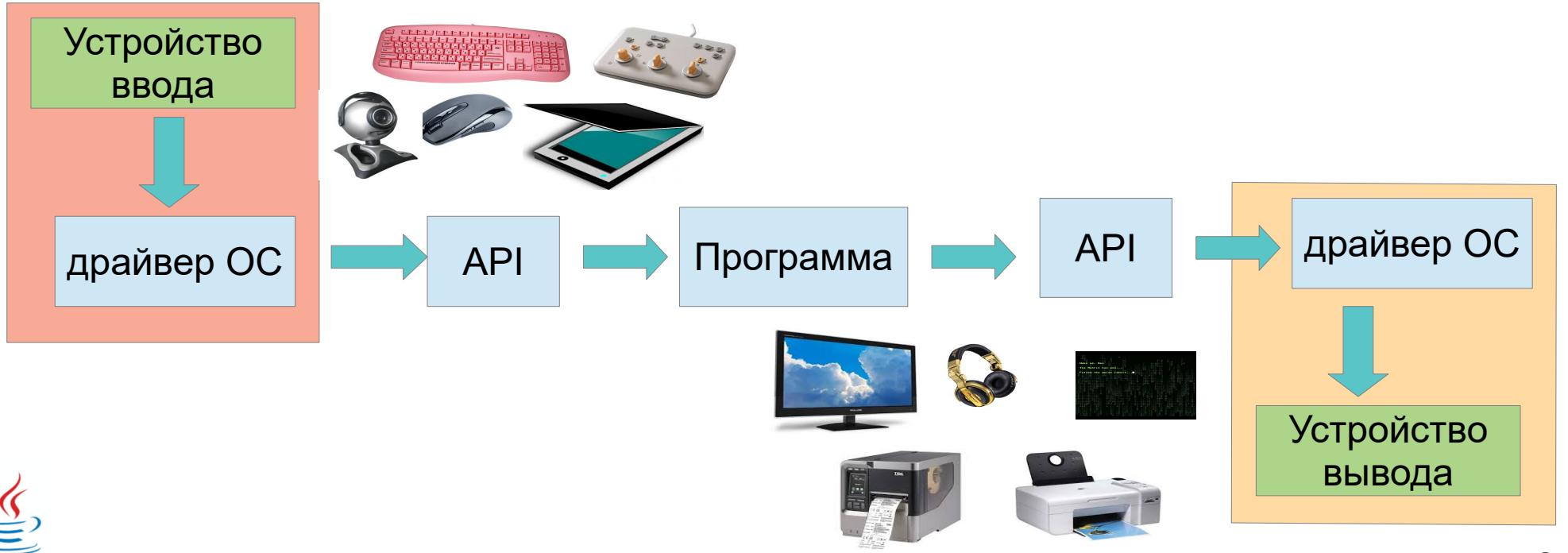
Ввод-вывод

Устройство ввода → драйвер ОС → API → Программа → API → драйвер ОС → Устройство вывода

Java

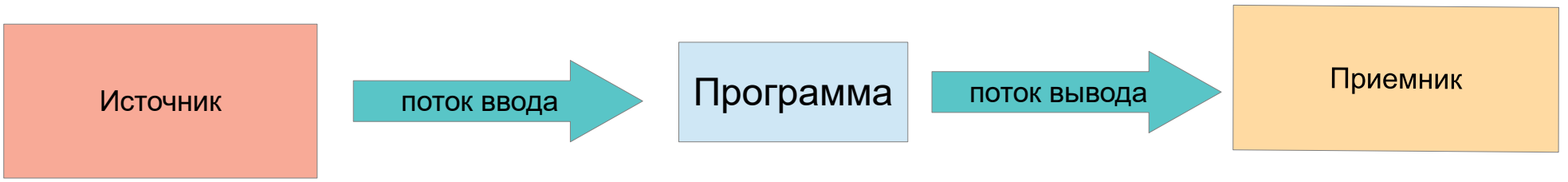
Абстракция ввода-вывода

- ☑ Потоки ввода-вывода
- ☑ Источник данных и приемник данных



Абстракция ввода-вывода

- ☑ Потоки ввода-вывода
- ☑ Источник данных и приемник данных



Java

Пакет java.io

1) Байтовые и символьные потоки данных (I/O streams)
   - Поток — последовательность данных (байтов, символов, примитивных типов, объектов)
   - Поток ввода — для чтения данных из источника
   - Поток вывода — для записи данных в приемник
2) Старый интерфейс работы с файлами — класс File

Байты и символы

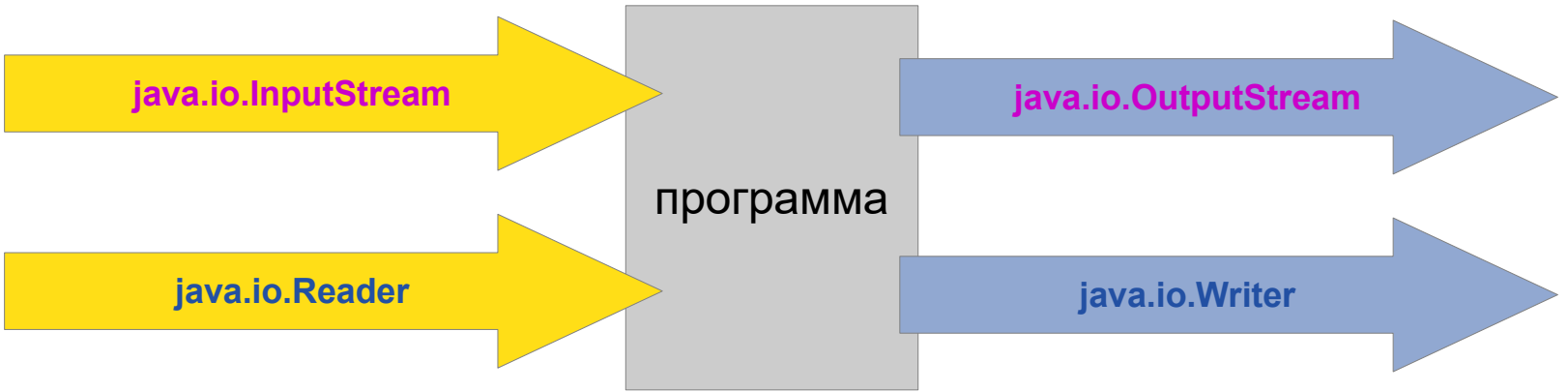
УНИВЕРСИТЕТ ИТМО

```
Integer number = 51966;
00000000 00000000 11001010 11111110 // big-endian
11111110 11001010 00000000 00000000 // little-endian
number.toString()      // "51966" =
'5' '1' '9' '6' '6'
00000000 00110101 00000000 00110001 00000000 00111001
00000000 00110110 00000000 00110110
number.toHexString()  //  "cafe"
'c' 'a' 'f' 'e'
00000000 01100011 00000000 01100001 00000000 01100110
00000000 01100101
UTF-8 – BOM (byte order mark)    '\uFEFF'
```

Java

Потоки ввода-вывода

- ☑ Базовые абстрактные классы для потоков
- ☑ Ввода и вывода
- ☑ Байтовые и символьные



Java

java.io.InputStream

- ☑ abstract **int** read()
  - прочитанный байт
  - -1 (конец потока)

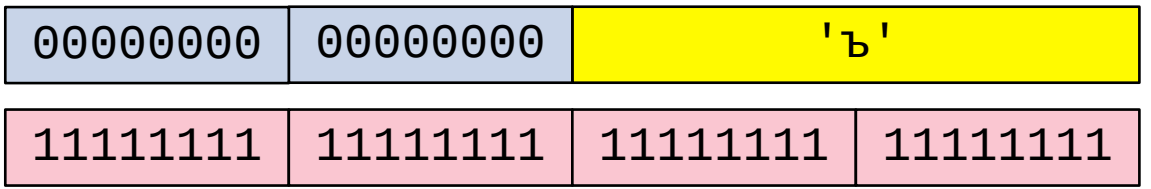
00000000	00000000	00000000	10100111
11111111	11111111	11111111	11111111

- ☑ int read(byte[] buf, int off, int len)

- ☑ int available()
- ☑ long skip(long n)
- ☑ void close()
- ☑ boolean markSupported()
- ☑ void mark(int limit)
- ☑ void reset()

java.io.Reader

- ☑ abstract **int** read(char[] buf, int off, int len)
  - количество байт
  - -1 (конец потока)
- ☑ int read()
  - read(buf, 0, 1)



- ☑ int available()
- ☑ long skip(long n)
- ☑ void close()
- ☑ boolean markSupported()
- ☑ void mark(int limit)
- ☑ void reset()

java.io.OutputStream

- ☑ abstract void write(**int** b)
  - записываемый байт

- ☑ void write(byte[] buf, int off, int len)

- ☑ void flush()
- ☑ void close()

java.io.Writer

- ☑ abstract write(char[] buf, int off, int len)
- ☑ void write(int c)
  - write(buf, 0, 1)

- ☑ void flush()
- ☑ void close()
- ☑ Writer append(int c)
- ☑ Writer append(CharSequence cs)

Interface Flushable

- ☑ flush() - очистка внутреннего кэша или буфера
- ☑ данные сливаются в приемник
  - После выполнения метода внутренние буферы и кэши должны быть пустыми, данные переданы операционной системе для записи

Интерфейс Closeable

- ☑ close() throws IOException - освобождение ресурса

```
try {
    InputStream ins = new InputStream();
    ins.read();
} finally {
    if (ins != null) ins.close();
}
```

Интерфейс AutoCloseable

- ☑ Closable extends AutoCloseable
- ☑ close() вызывается автоматически
- ☑ блок try с ресурсом

```
try(InputStream ins = new InputStream()) {
    ins.read();
}
```

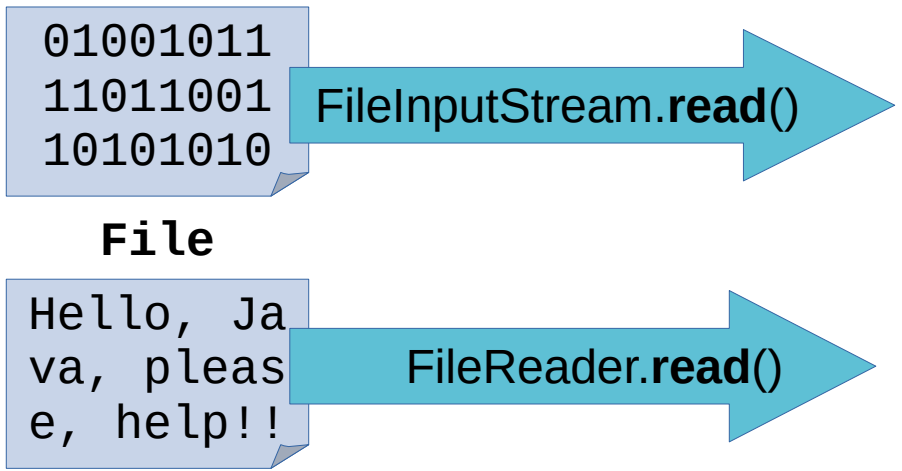
Java 9

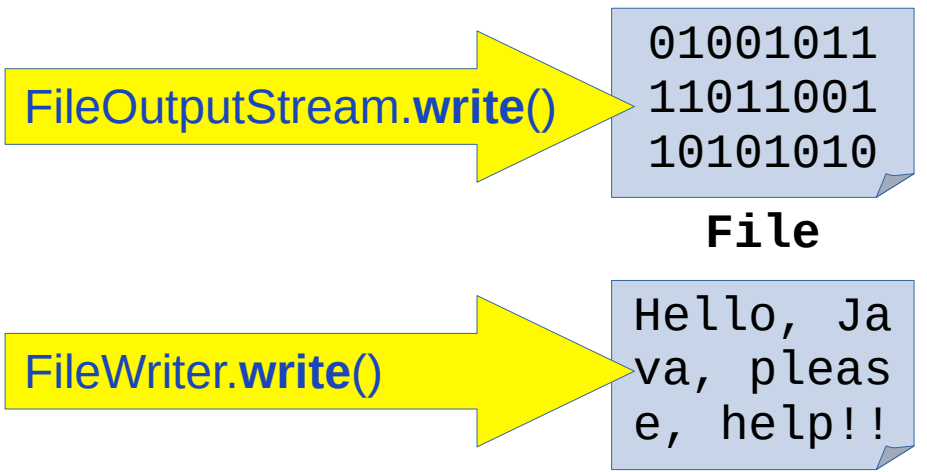
```
InputStream ins = new InputStream();
try(ins) {
    ins.read();
}
```

Специализированные потоки - File

УНИВЕРСИТЕТ ИТМО

- ☑ FileOutputStream
- ☑ FileWriter
- ☑ FileInputStream
- ☑ FileReader





Java

Чтение и запись с помощью потоков

Java 9

```
FileInputStream in = new FileInputStream("in.bin");
FileOutputStream out = new FileOutputStream("out.bin");
try (in, out) {
  int c;
  while ((c = in.read()) != -1) {
    out.write(c);
  }
  out.flush();
} catch (IOException e) {
  System.err.println(e);
}
```

Java

Чтение и запись с помощью потоков

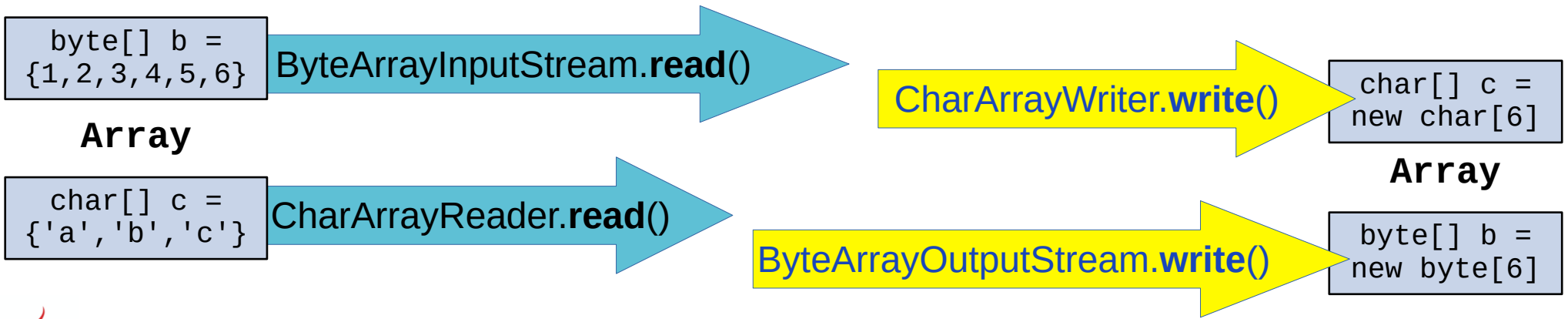

```
try (FileReader in = new FileReader("in.txt");
     FileWriter out = new FileWriter("out.txt")) {
  int c;
  while ((c = in.read()) != -1) {
    out.write(c);
  }
  out.flush();
} catch (IOException e) {
  System.err.println(e);
}
```

Java

Специализированные потоки - Array

УНИВЕРСИТЕТ ИТМО

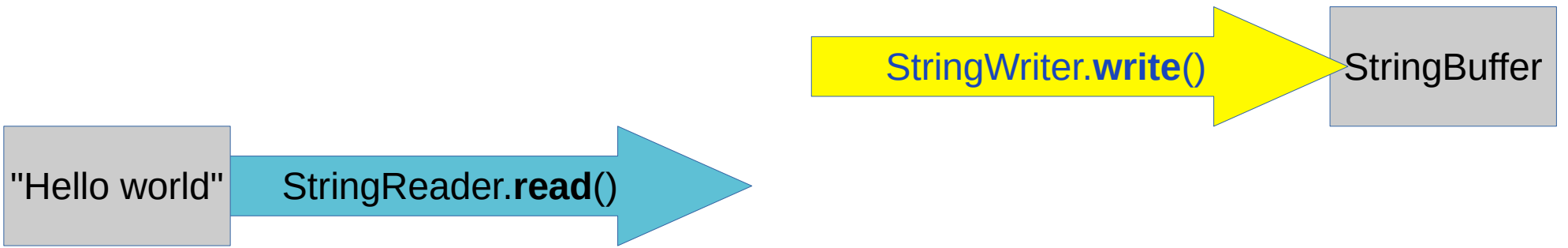
- ☑ ByteArrayOutputStream
  - toByteArray()
- ☑ CharArrayWriter
  - toCharArray()
- ☑ ByteArrayInputStream
- ☑ CharArrayReader



Java

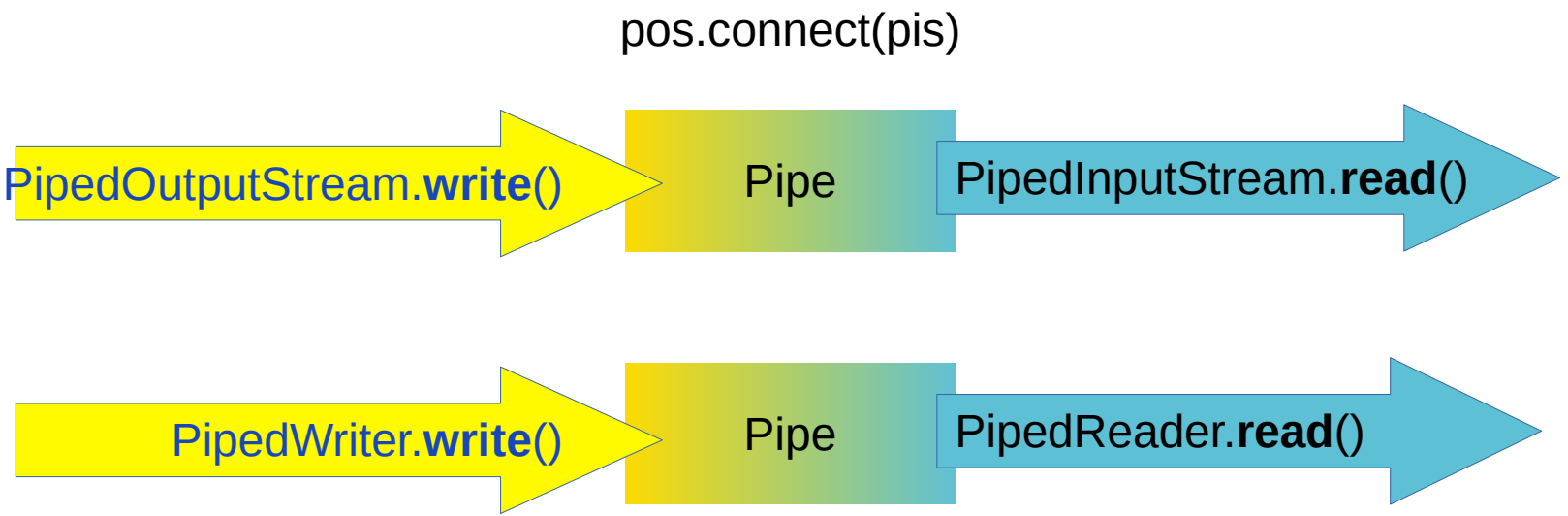
Специализированные потоки - String

- ☑ StringWriter
  - StringBuffer getBuffer()
  - String toString()
- ☑ StringReader



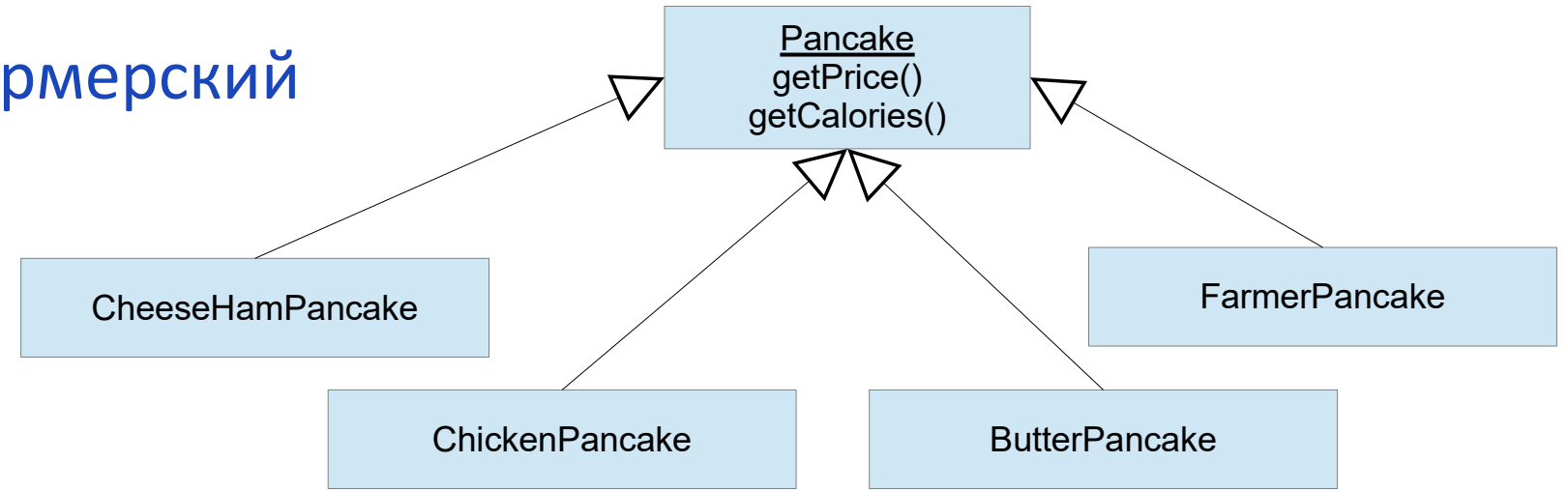
Специализированные потоки - Pipe

- ☑ PipedOutputStream
- ☑ PipedInputStream
- ☑ PipedWriter
- ☑ PipedReader



Шаблон Decorator - Теремок

- ☑ Блин с ветчиной и сыром
- ☑ Блин с куриной грудкой
- ☑ Блин с маслом
- ☑ Блин Фермерский



Шаблон Decorator - Теремок

- ☑ Блин с ветчиной и сыром
- ☑ Блин с куриной грудкой
- ☑ Блин с маслом
- ☑ Блин Фермерский
- ☑ Добавки
  - Огурцы
  - Лук-фри
  - Картофельное пюре
  - Сметана

Шаблон Decorator - Теремок

УНИВЕРСИТЕТ ИТМО

- ☑ Блин с ветчиной и сыром
- ☑ Блин с куриной грудкой
- ☑ Блин с маслом
- ☑ Блин Фермерский

- ☑ Добавки
  - Огурцы 30 руб.
  - Лук-фри
  - Картофельное пюре
  - Сметана

```
class CucumberPancake extends Pancake {
  Pancake base;
• CucumberPancake(Pancake p) { base = p; }
• double getPrice() { return base.getPrice() + 30); }
}
```

Шаблон Decorator - Теремок

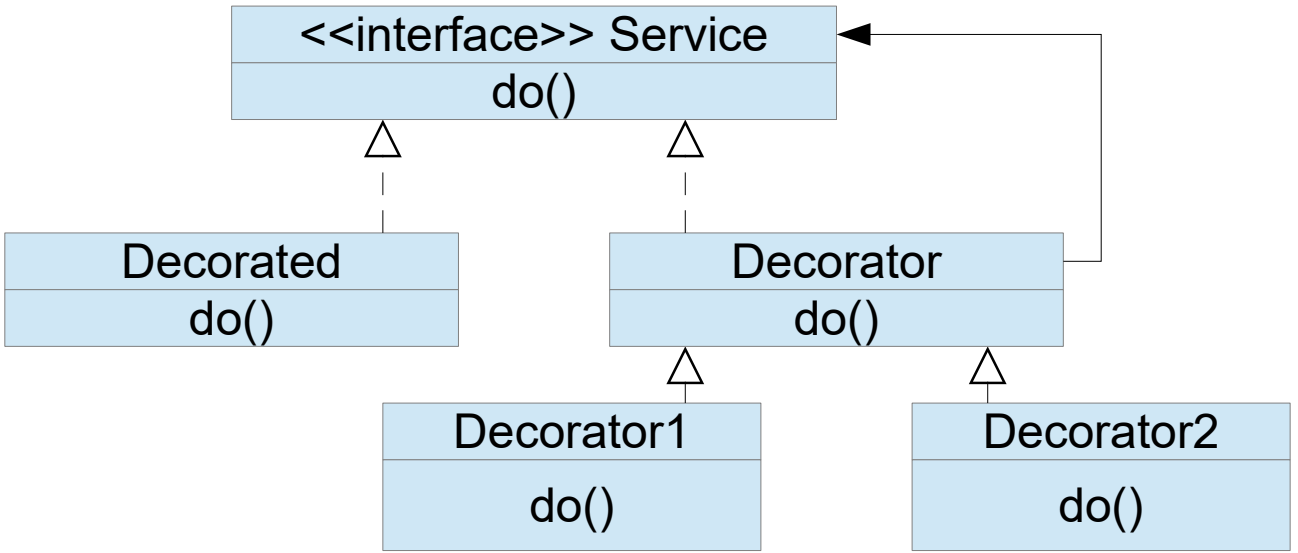
- ☑ Блин с ветчиной и сыром
- ☑ Блин с куриной грудкой
- ☑ Блин с маслом
- ☑ Блин Фермерский
- ☑ Огурцы
- ☑ Лук-фри
- ☑ Картофельное пюре
- ☑ Сметана

```
Pancake pc =
    new MashedPotatoPancake(
        new CucumberPancake(
            new CheeseHamPancake()));

pc.getPrice();
```

Шаблон Decorator

- ☑ Динамически добавляет объекту функциональность

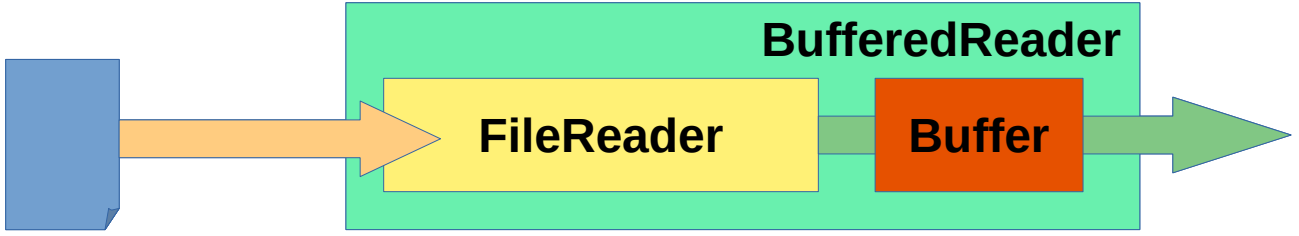


Потоки-фильтры - декораторы

- ☑ FilterInputStream(InputStream)
- ☑ FilterOutpurStream(OutpurStream)
- ☑ FilterReader(Reader)
- ☑ FilterWriter(Writer)
  - исходный поток = аргумент конструктора
  - поток-фильтр = декоратор

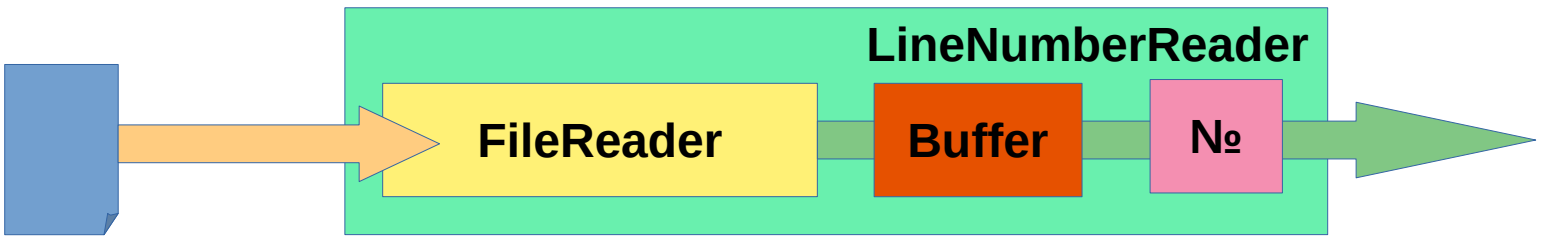
Потоки-фильтры - Buffered

- ☑ BufferedInputStream
- ☑ BufferedOutpurStream
- ☑ BufferedReader
- ☑ BufferedWriter
- ☑ Буфер для повышения производительности
- ☑ Возможность построчной работы



Потоки-фильтры - LineNumber

- ☑ ~~LineNumberInputStream~~
- ☑ LineNumberReader extends BufferedReader
  - getLineNumber()
  - setLineNumber(int) - меняет только номер, не саму строку



Чтение и запись с помощью потоков


```
try (LineNumberReader in = new LineNumberReader(
                                new FileReader("in.txt"));
     BufferedWriter out = new BufferedWriter(
                                new FileWriter("out.txt")) {
  String line;
  while ((line = in.readLine()) != null) {
    out.write(in.getLineNumber() + ": " + line);
    out.newLine();
  }
  out.flush();
} catch (IOException e) {
  System.err.println(e);
}
```

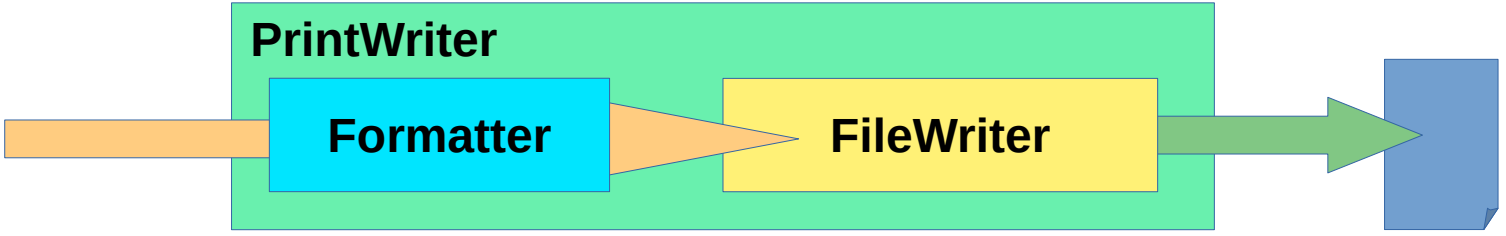
Java

PrintStream, PrintWriter

- ☑ PrintStream, PrintWriter
- ☑ print, println — один аргумент
- ☑ printf, format — форматная строка + аргументы

```
format("%c = %2$+9.7f", 'π', Math.PI);

π = +3,1415927
```



PrintStream, PrintWriter

☑ %[индекс$][флаги][размер][.точность]формат

- ☑ %b - boolean
- ☑ %h - hashcode
- ☑ %t = time/date
- ☑ %% - %
- ☑ %n - newline
- ☑ %d - decimal
- ☑ %o - octal
- ☑ %h - hex
- ☑ %s - string
- ☑ %c - char
- ☑ %f - float
- ☑ %e - exponent
- ☑ %g - %e / %f
- ☑ %a - hex float

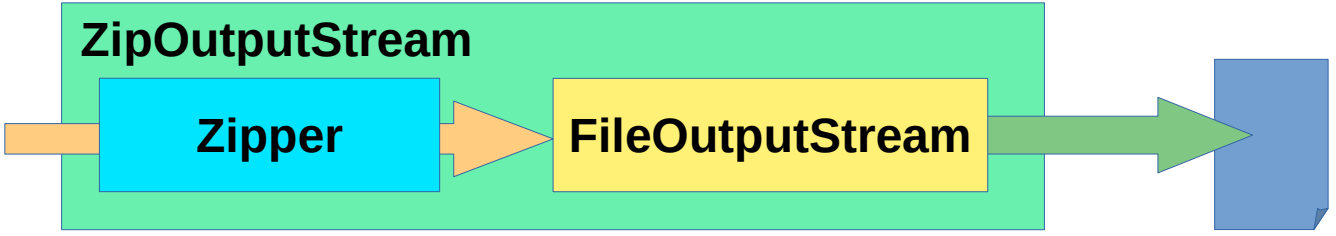
- ☑ индекс – номер аргумента
- ☑ флаги – зависят от формата
- ☑ размер – количество символов
- ☑ точность – после запятой

```
format("%c = %2$+9.7f", 'π', Math.PI);

π = +3,1415927
```

Потоки-фильтры - Inflater/Deflater

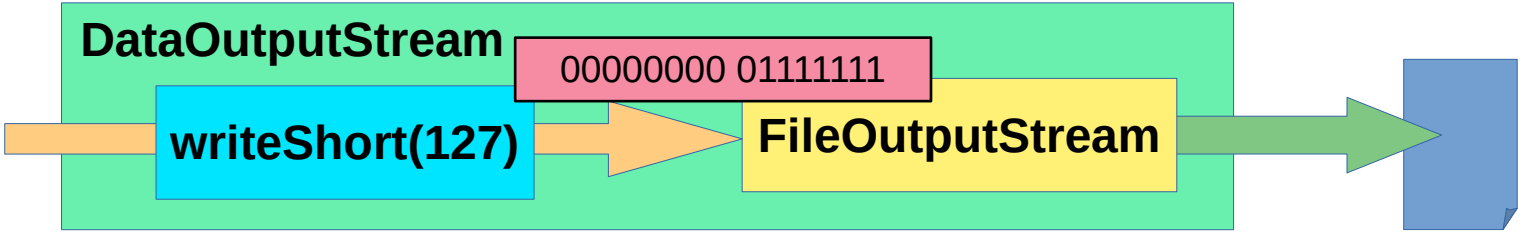
- ☑ потоки с компрессией
- ☑ InflaterInputStream
  - GzipInputStream
  - ZipInputStream
    - JarInputStream
- ☑ DeflaterOutputStream
  - GzipOutputStream
  - ZipOutputStream
    - JarOutputStream



Java

DataInputStream, DataOutputStream

- ☑ DataInputStream, DataOutputStream
- ☑ примитивы и строки ↔ байты
- ☑ Последовательность при чтении - такая же как при записи:
- ☑ DataOutputStream.writeInt(int), DataInputStream.readInt()
  - byte, short, long, float, double, chaar, boolean
  - writeUTF(String), readUTF()



Java

Сериализация объектов

- ☑ Сериализация — запись объектов в виде потока байтов
- ☑ Классы ObjectOutputStream, ObjectInputStream
- ☑ Интерфейс-метка Serializable
- ☑ При записи объекты записываются с порядковым номером (serial number)
- ☑ Объекты записываются в поток иерархически (deep copy)
- ☑ Объект записывается в поток только один раз, потом используется ссылка на номер объекта
- ☑ При чтении одного и того же объекта из одного потока, он восстанавливается один раз
- ☑ При чтении объекта из двух потоков, объект восстанавливается дважды

ObjectOutputStream, ObjectInputStream

- ☑ Те же методы, что и у DataOutputStream, DataInputStream
  - + writeObject(Object)
  - + Object readObject()
- ☑ Методы записывают/читают:
  - класс объекта
  - сигнатуру класса (зависит от имени и модификаторов класса, интерфейсов, типов и имен полей, имен и сигнатур конструкторов и методов)
  - сигнатура класса может быть записана в поле serialVersionUID
  - значения нестатических и непереходных полей данного класса, и всех его суперклассов
  - не сериализуются поля с модификаторами static и transient

Проблемы стандартной сериализации

1) Сериализованный объект зависит от внутренней структуры класса
2) Объекты могут создаваться в обход конструкторов
3) Потенциальные проблемы совместимости версий
4) Возможные проблемы с:
    1) продолжительностью сериализации
    2) необходимой памятью на сериализацию
    3) необходимой глубиной стека

Замена стандартной сериализации

- ☑ Задать несериализуемым полям модификатор transient
- ☑ Реализовать методы в сериализуемом классе
  - private void writeObject(ObjectOutputStream os)
  - private void readObject(ObjectInputStream is)
- ☑ внутри этих методов вызываются методы
  - os.defaultWriteObject()
  - is.defaultReadObject()
- ☑ Суперкласс не трогаем

Замена стандартной сериализации

- ☑ Реализуем интерфейс Externalizable
- ☑ реализуем методы (полный контроль сериализации)
  - writeExternal(ObjectOutput o)
  - readExternal(ObjectInput i)
- ☑ Необходимо самостоятельно обрабатывать суперкласс
- ☑ При сериализации вместо стандартного механизма будет вызван метод writeExternal, при десериализации - readExternal

☑ Потоки-фильтры - байт ↔ символ

- ☑ Чтение-запись в других кодировках
- ☑ InputStreamReader extends Reader
  - байты в символы
  - new InputStreamReader(InputStream in, *кодировка*)
- ☑ OutputStreamWriter extends Writer
  - символы в байты
  - new OutputStreamWriter(OutputStream out, *кодировка*)

Java

Scanner

☑ Конструкторы

- Scanner(File)
- Scanner(Path)
- Scanner(InputStream)
- Scanner(Readable)
- Scanner(String)

☑ Методы

- boolean hasNext(), String next()
- boolean hasNextInt(), nextInt()
  - ….long, byte, short, float, double, boolean, char
- String nextLine(), nextPattern()
- void useDelimiter(String)
- void useRadix(int)

Консольный ввод-вывод

- ☑ Стандартные потоки ввода-вывода
  - InputStream System.in
  - PrintStream System.out
  - PrintStream System.err
- ☑ java.io.Console
  - Console c = System.console()
  - cons.readLine()
  - cons.readPassword()
  - cons.printf()
  - cons.format()

УНИВЕРСИТЕТ ИТМО

Программирование. 2 семестр

Вспомогательные классы

ITMOre than a UNIVERSITY

Класс java.util.Random

- ☑ public Random()
- ☑ public Random(long seed)
- ☑ nextInt(), nextDouble, nextLong, nextGaussian
- ☑ IntStream ints()
- ☑ LongStream longs()
- ☑ DoubleStream doubles()

Регулярные выражения

- ☑ Класс Pattern — представляет регулярное выражение
- ☑ Класс Matcher — движок, проверяющий соответствие

```
String regex = "a*b";
Pattern p = Pattern.compile(regex);
Matcher m = p.matcher("aaabbb");
boolean b = m.matches();

boolean b = Pattern.matches(regex, "aaabbb");
```

Регулярные выражения


```
x - x
\\ - \
\0nnn - 8-ричный код (байт)
\xhh  - 16-ричный код (байт)
\uhhhh - 16-ричный код (Unicode)
\t - tab
\n - newline
\r - carriage-return
```

Регулярные выражения


```
.
[abcde]
[^abcde]
[a-zA-Z]
[a-z&&[^ij]]

\d  [0-9]          \D [^0-9]
\w  [a-zA-Z_0-9]   \W
\s  [ \t\n\r]      \S
\h \v \H \V
```

Регулярные выражения

УНИВЕРСИТЕТ ИТМО

```
\p{Punct} = [!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~]
\p{Digit} = [0123456789]
\p{XDigit} = [0123456789abcdefABCDEF]
\p{Lower} = [abcdefghijklmnopqrstuvwxyz]
\p{Upper} = [ABCDEFGHIJKLMNOPQRSTUVWXYZ]
\p{Alpha} = [\p{Lower}\p{Upper}]
\p{Alnum} = [\p{Alpha}\p{Digit}]
\p{Graph} = [\p{Alnum}\p{Punct}]
\p{Print} = [ \p{Graph}]
\p{Blank} \p{Space} \p{Cntrl} \p{ASCII}
```

Регулярные выражения

УНИВЕРСИТЕТ ИТМО

^X	X$				
XY	X|Y				
(X)	(?:X)	\1 \2			
X{2}	X{2,}	X{2,3}			
X{0,1}	X{0,}	X{1,}			
X?	X*	X+	Greedy	a.*b	abaaab
X??	X*?	X+?	Reluctant	a.*?b	abaaab
X?+	X*+	X++	Posessive	a.*+b	abaaab

```
\w+@(\w+\.)+\w+

info@cs.itmo.ru
```

Java

Запись и чтение CSV, JSON, XML

- Запись
  - println() / format()
- Чтение
  - Scanner, StreamTokenizer, Regex
- CSV
  - ! специальные символы, разделители, переносы строк

Запись и чтение CSV, JSON, XML

- ☑ Binding
  - XML/JSON <-> Java Object (целиком)
  - низкая скорость, настройка на класс, много памяти, простая обработка
  - JAXB
- ☑ DOM (Document Object Model)
  - XML/JSON <-> Object Tree (Graph)
  - Node, Element
  - универсальная модель, много памяти, средняя сложность
  - XML DOM
- ☑ Event/Stream
  - element start / element end
  - высокая скорость, мало памяти, сложнее обработка
  - SAX, StAX

Запись и чтение CSV, JSON, XML

- Стандартные библиотеки java
  - XML
    - Java API for XML Processing (JAXP)
    - javax.xml.parsers
    - org.w3c.dom
    - org.xml.sax
    - javax.xml.stream
    - javax.xml.bind
  - JSON
    - javax.json.* (Java EE)

Запись и чтение CSV, JSON, XML

- Сторонние библиотеки
  - Apache Commons CSV, OpenCSV
  - GSON, Jackson
  - SimpleXML, XStream, JacksonXML

Javadoc


```
/**
 * This is my class
 * @author I
 */
public class MyClass {
/**
 * This is my method
 * @params x just x
 * @return double x
 */
  public int doubleX(int x) {
    return x * 2;
  }
}

javadoc -d doc *.java
```