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

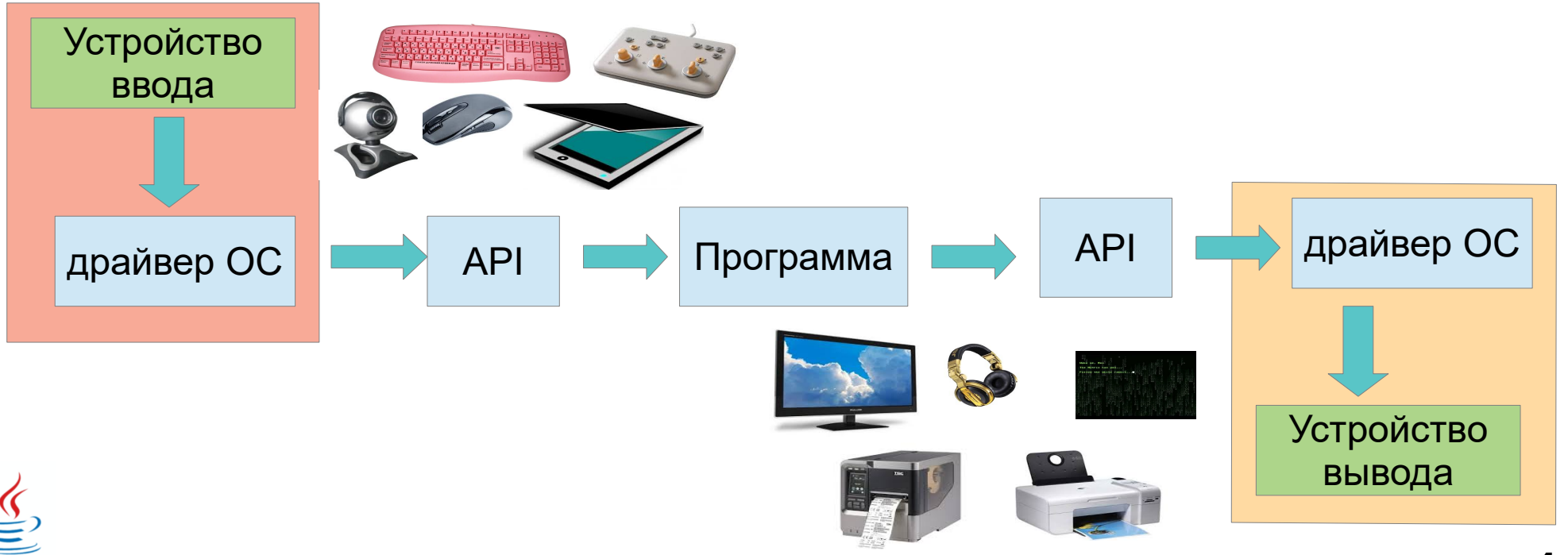
Ввод-вывод

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

Java

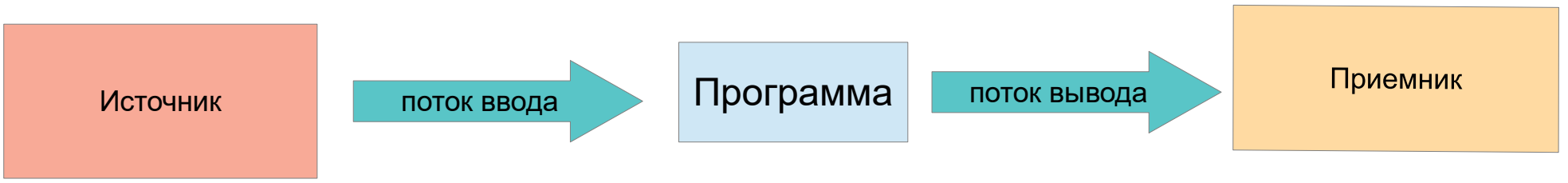
Абстракция ввода-вывода

- ☑ 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

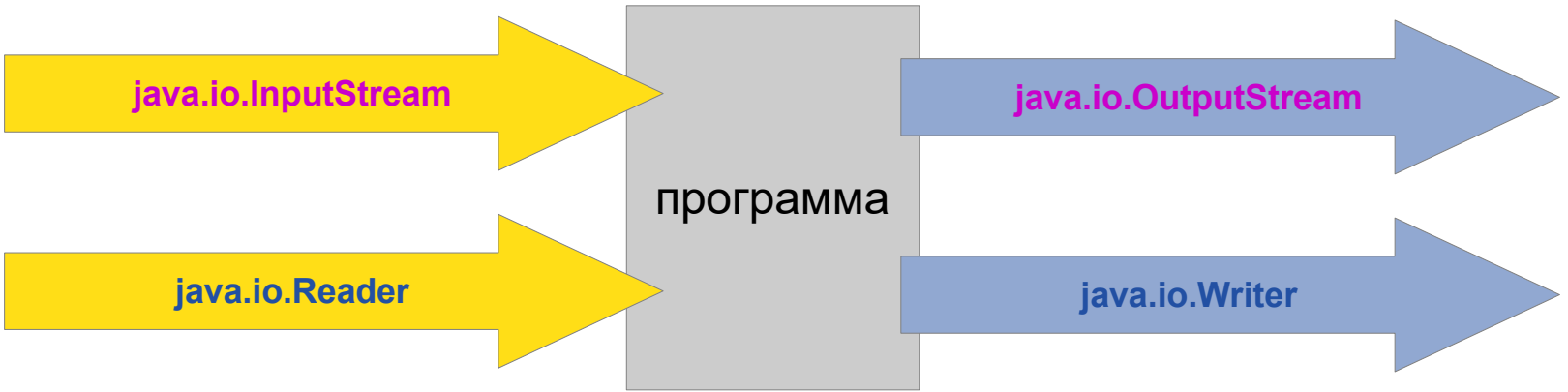
Байты и символы


```
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)

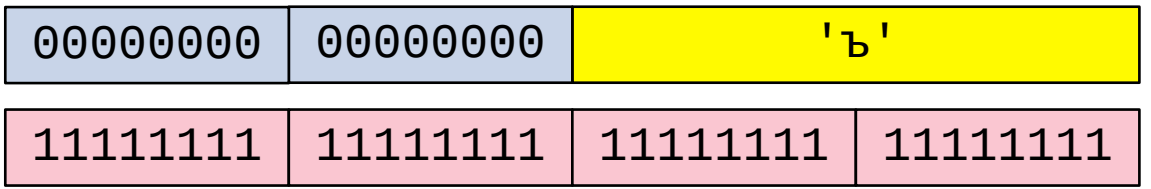
00000000	00000000	00000000	10100111
11111111	11111111	11111111	11111111

- ☑ 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

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();
}
```

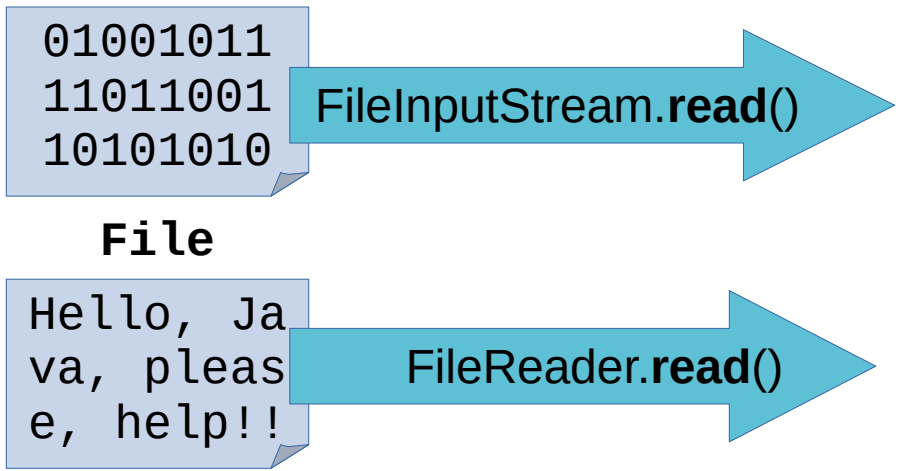
Java 9

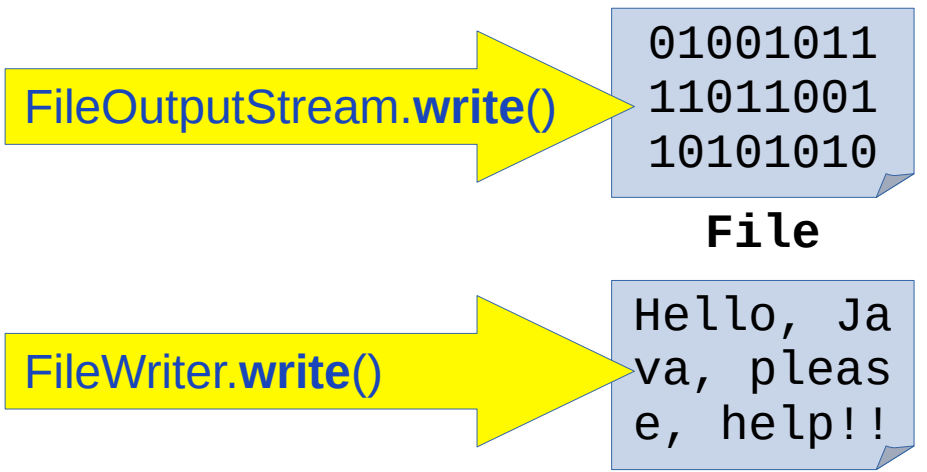
```
InputStream ins = new InputStream();
try(ins) {
    ins.read();
}
```

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

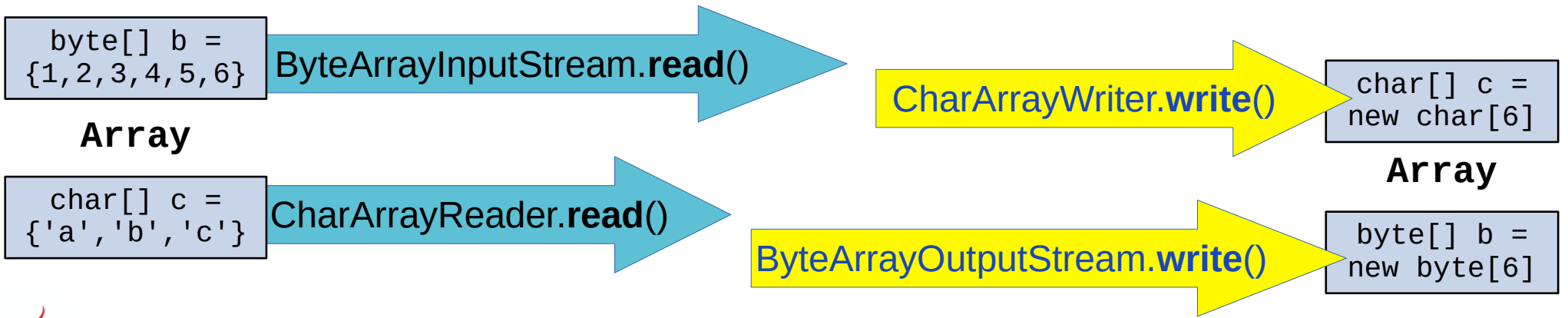

```
try (FileReader in = new FileReader("in.txt");
     FileWriter out = new FileWriter("out.txt")) {
  int c;
  while ((c = in.read()) != -1) {
    out.write(c);
  }
  out.flush();
} catch (IOException e) {
  System.err.println(e);
}
```

Java

Специализированные потоки - Array

УНИВЕРСИТЕТ ИТМО

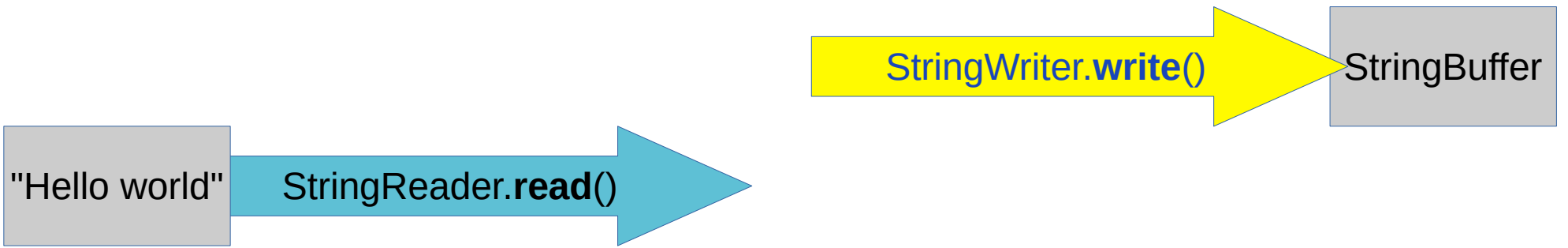
- ☑ ByteArrayOutputStream
  - toByteArray()
- ☑ CharArrayWriter
  - toCharArray()
- ☑ ByteArrayInputStream
- ☑ CharArrayReader



Java

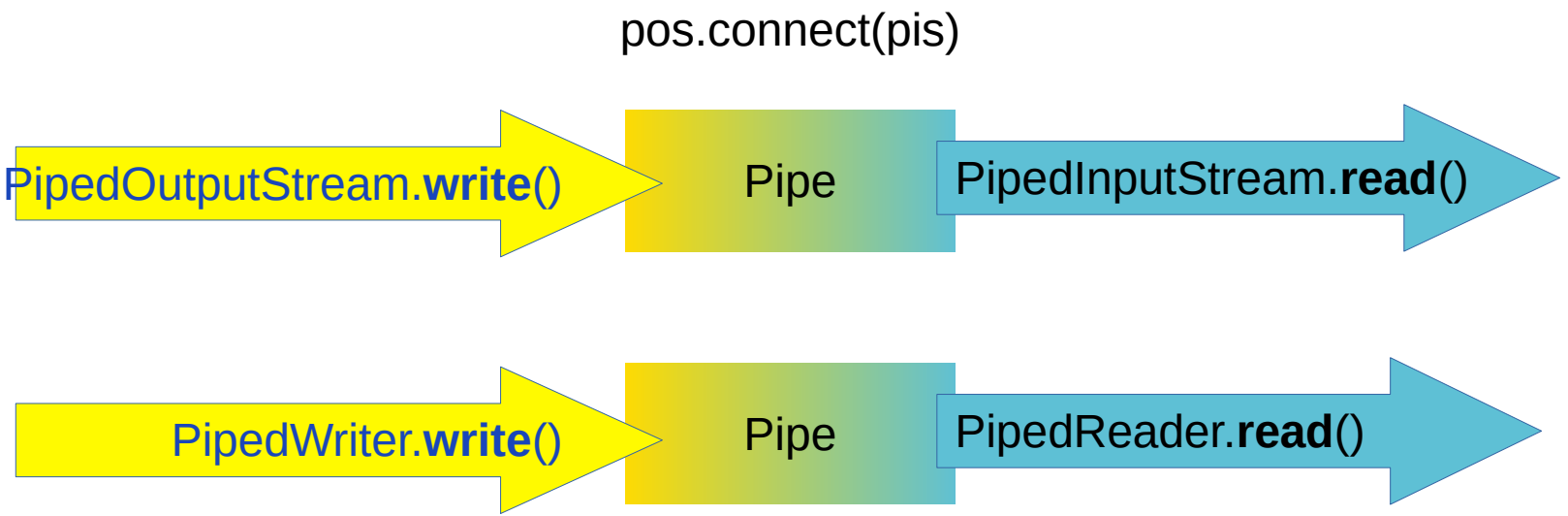
Специализированные потоки - String

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



Специализированные потоки - Pipe

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



УНИВЕРСИТЕТ ИТМО

Программирование. 2 семестр

Шаблоны проектирования (введение)

ITMO more than a UNIVERSITY

Шаблоны проектирования

УНИВЕРСИТЕТ ИТМО

- ☑ Кто-то когда-то уже делал что-то похожее
- ☑ Пришлось вносить изменения - возникли проблемы
- ☑ Нужно сразу было делать по-другому!

- ☑ GoF Book
  - Gang of Four

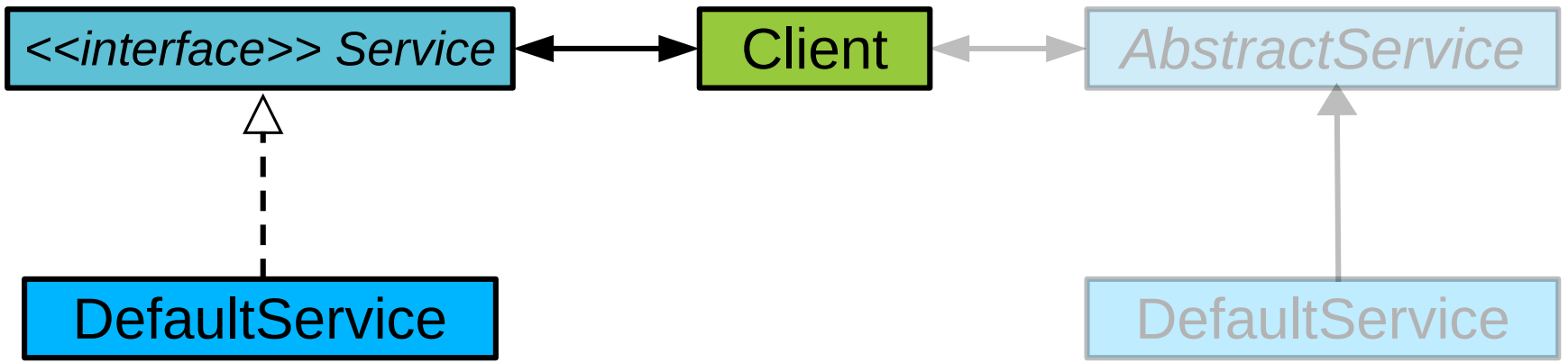
Java

Основные принципы проектирования

- ☑ Повторное использование кода
  - DRY - Don't repeat yourself
  - стандартные библиотеки
  - фреймворки
- ☑ Расширяемость
  - Учет возможных будущих изменений

Интерфейс / Абстрактный суперкласс

- ☑ Независимость от реализации - больше гибкость и универсальность
- ☑ Унификация поведения потомков - проще реализация

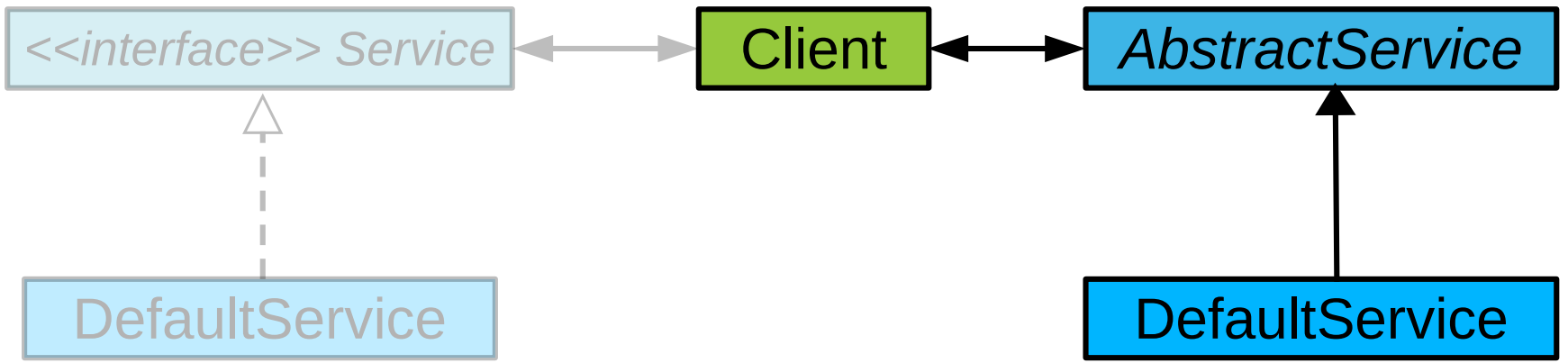


Java

Интерфейс / Абстрактный суперкласс

УНИВЕРСИТЕТ ИТМО

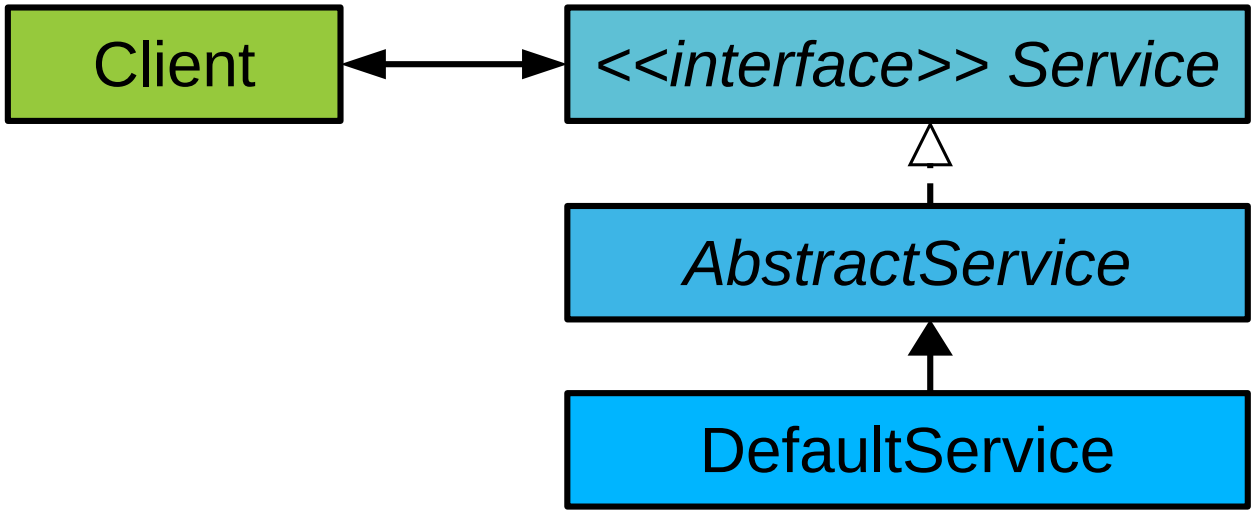
- ☑ Независимость от реализации - больше гибкость и универсальность
- ☑ Унификация поведения потомков - проще реализация



Java

Интерфейс + абстрактный суперкласс

- ☑ Общие свойства - АБСТРАКТНЫЙ КЛАСС
- ☑ Взаимодействие с абстракцией - ИНТЕРФЕЙС
- ☑ Set<T> ⇠ AbstractSet<T> ⇦ HashSet<T>



Java

Наследование и делегирование

- ☑ Наследование - подкласс
  - собака - подкласс животного
  - статическое отношение **is-a** (Dog is an animal)

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

Наследование и делегирование

- ☑ Наследование - подкласс
  - собака - подкласс животного
  - статическое отношение **is-a** (Dog is an animal)
- ☑ Student и Person
  - студент - **роль**

```
class Dog extends Animal  { }

class Person { }

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

Java

Наследование и делегирование

- ☑ Наследование - подкласс
  - собака - подкласс животного
  - статическое отношение **is-a** (Dog is an animal)
- ☑ Student и Person
  - студент - **роль**
  - роль может меняться
- ☑ Делегирование!

```
class Dog extends Animal  { }

class Person {
  public getName() { }
}

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

Java

Наследование и композиция

- ☑ Наследование - подкласс
  - собака - подкласс животного
  - статическое отношение **is-a** (Dog is an animal)
- ☑ Композиция
  - "часть-целое"
  - отношение **has-a**

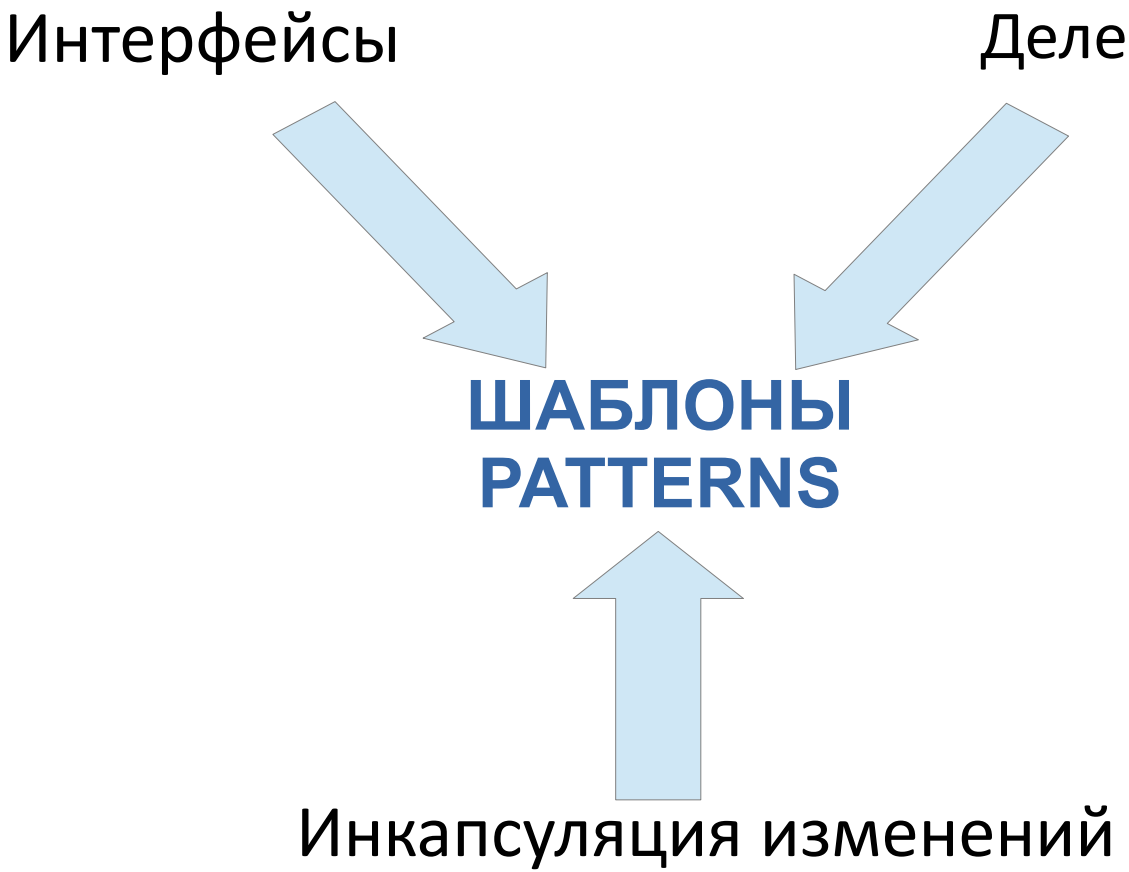
```
class Dog extends Animal  { }

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

Java

Инкапсуляция изменений

- ☑ Отделить изменяющееся от постоянного
- ☑ Инкапсулировать изменяющееся

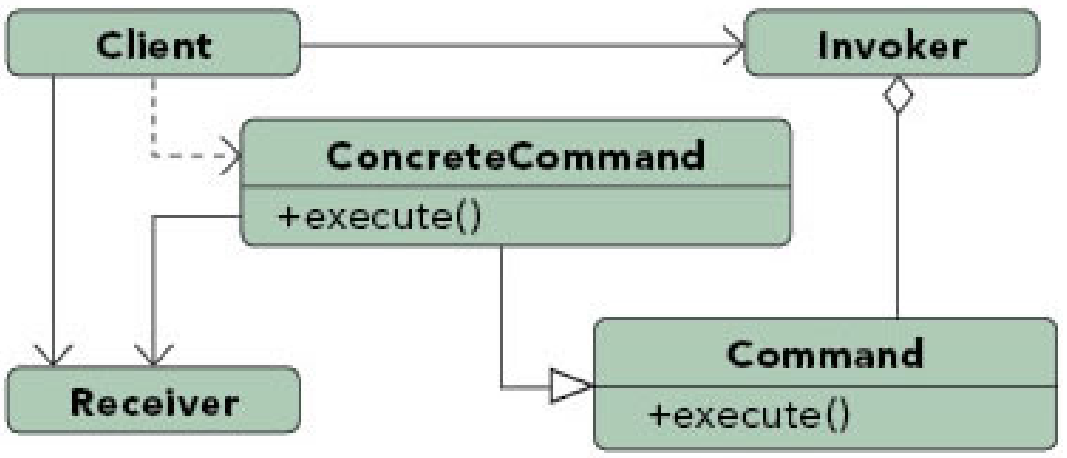
Как справиться с изменениями?



Java

Шаблон Команда (Command)

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



Лаба 5 - плохой вариант

УНИВЕРСИТЕТ ИТМО

```
Scanner sc = new Scanner(System.in);
String line = sc.next();
String[] tokens = line.split(" ");
if (tokens[0].equals("help") {
    doHelp();
}
if (tokens[0].equals("add") {
    doAdd();
}
...

public void doHelp() {
    System.out.println("help - помощь");
    System.out.println("add - добавить элемент");
}
```

Лаба 5 - более правильный вариант

УНИВЕРСИТЕТ ИТМО

```
public interface Command {            // Abstract Command
    void execute();
}
public class CollectionManager {   // Receiver (исполнитель)
    List<Pokemon> pokemonList = new ArrayList<>();
    public add(Pokemon p) {
        pokemonList.add(p);
    }
}
public class AddCommand implements Command {  // Add Command
    CollectionManager cm;
    public void execute() {
        cm.add(pokemon);
    }
}
```

Лаба 5 - более правильный вариант

УНИВЕРСИТЕТ ИТМО

```
public class Invoker {
    ...
    Map<String, Command> commands = new HashMap<>();
    commands.put("help", new HelpCommand());
    commands.put("add", new AddCommand());
    ...
    Scanner sc = new Scanner(System.in);
    while (sc.hasNext()) {
        String line = sc.next();
        String[] tokens = line.split(" ");
        Command command = commands.get(tokens[0]);
        command.execute();
    }
```

Лаба 5 - более правильный вариант

УНИВЕРСИТЕТ ИТМО

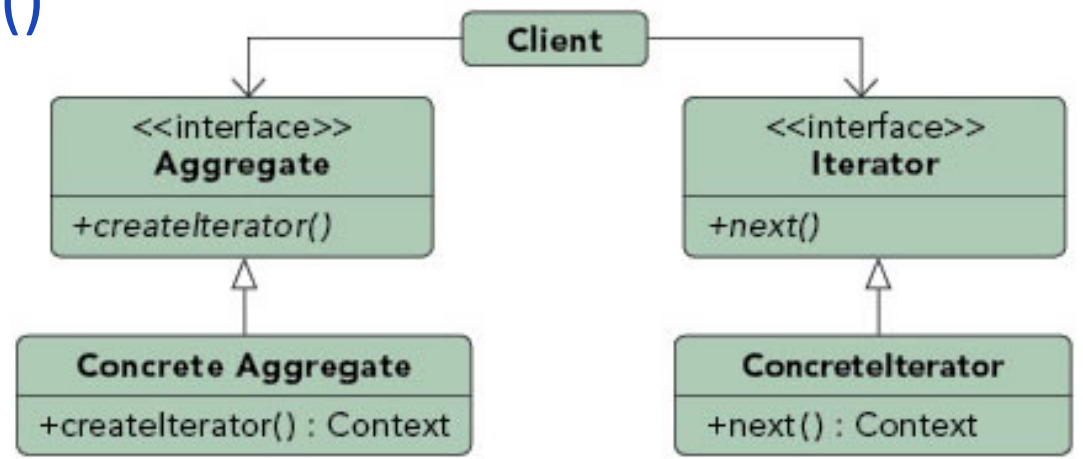
```
public interface Command {              // Abstract Command
    void execute();
    String descr();
}

public class HelpCommand implements Command {
    public String descr() { return "help - помощь"; }
    public void execute() {
        for (Command c : commands.values()) {
            System.out.println(c.descr());
        }
}
```

Виды шаблонов

- ☑ Порождающие
  - Singleton, Factory Method, Builder, ...
- ☑ Структурные
  - Adapter, Decorator, Proxy
- ☑ Поведенческие
  - Command, Iterator, Observer, Strategy

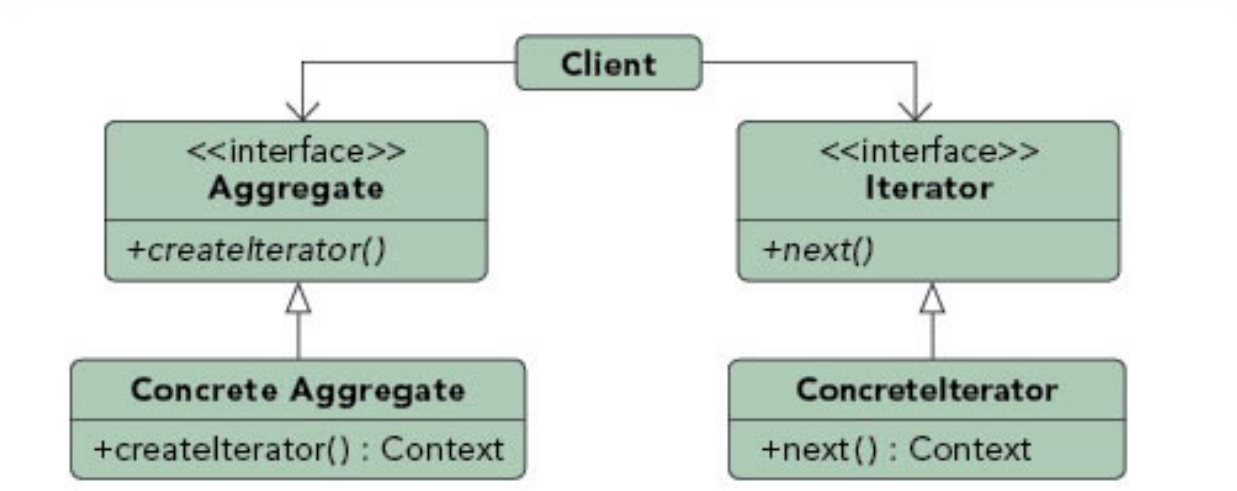
Шаблон Iterator

- ☑ Последовательный доступ к элементам
- ☑ Collection.iterator()
  - Iterator.hasNext(), .next()
- ☑ Scanner.hasNext(), .next()



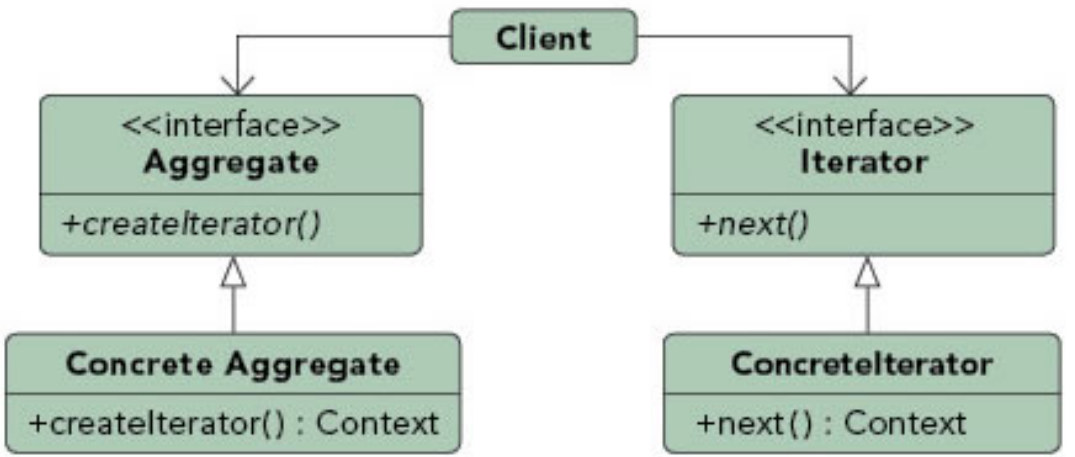
Шаблон Iterator

- ☑ Последовательный доступ к элементам коллекции
- ☑ Экскурсия: итератор - список достопримечательностей
  - Реальный гид
  - Аудиогид
  - Путеводитель



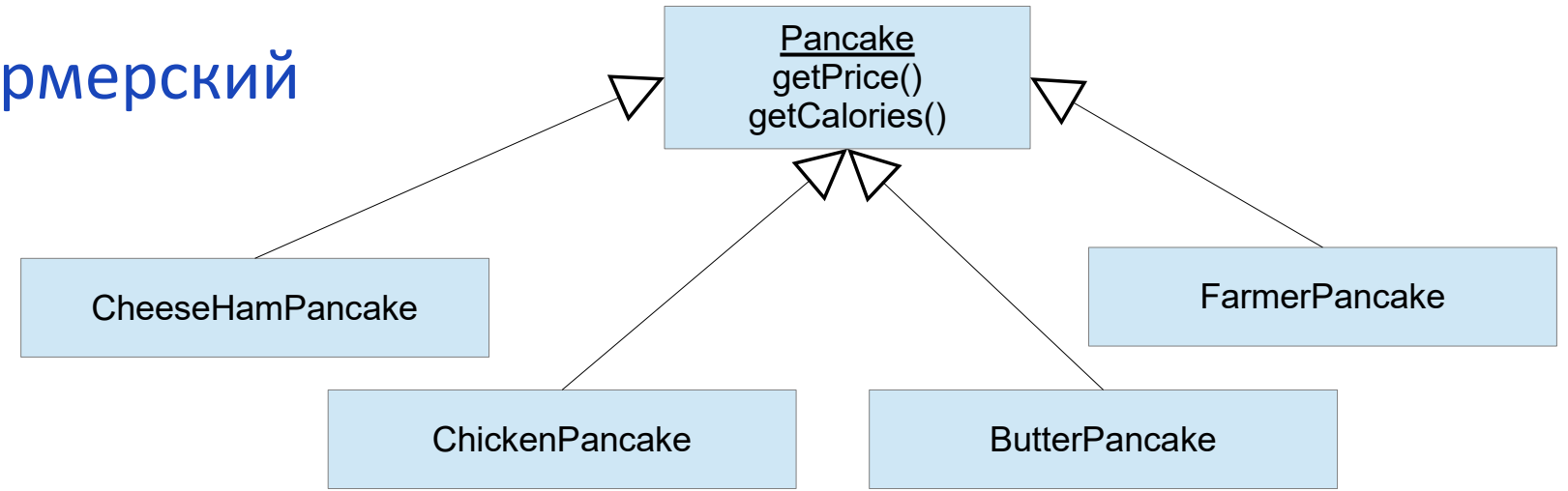
Iterator

- ☑ Универсальный доступ ко всем элементам коллекций
- ☑ Гибкая реализация обхода коллекций
- ☑ Более сложный вариант, чем простой цикл



Шаблон Decorator - Теремок

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



Шаблон Decorator - Теремок

УНИВЕРСИТЕТ ИТМО

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

- ☑ Добавки
  - Огурцы
  - Лук-фри
  - Картофельное пюре
  - Сметана

- ☑ ChickenPancakeWithPicklesAndMashedPotatoes
- ☑ FarmerPancakeWithSourCreamAndOnionAndPickles

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

Шаблон Decorator - Теремок

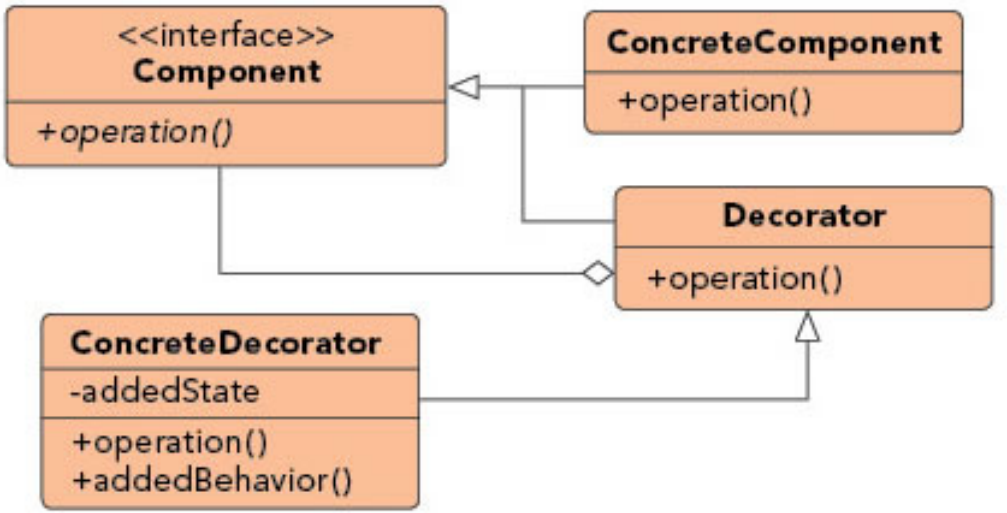
УНИВЕРСИТЕТ ИТМО

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

```
Pancake p =
    new MashedPotatoPancake(
        new PicklePancake(
            new CheeseHamPancake()));

p.getPrice();
```

Шаблон Decorator

- ☑ Позволяет добавлять функциональность динамически
- ☑ Вместо большой иерархии - несколько декораторов
- ☑ Сложность конфигурирования



УНИВЕРСИТЕТ ИТМО

Программирование. 2 семестр

Ввод-вывод (продолжение)

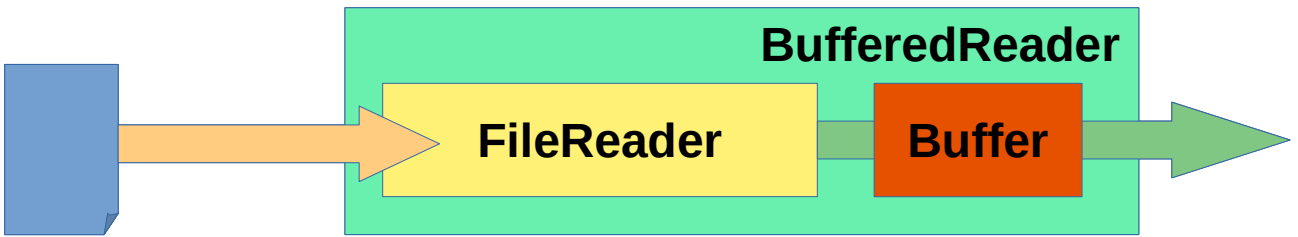
ITMO more than a UNIVERSITY

Потоки-фильтры - декораторы

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

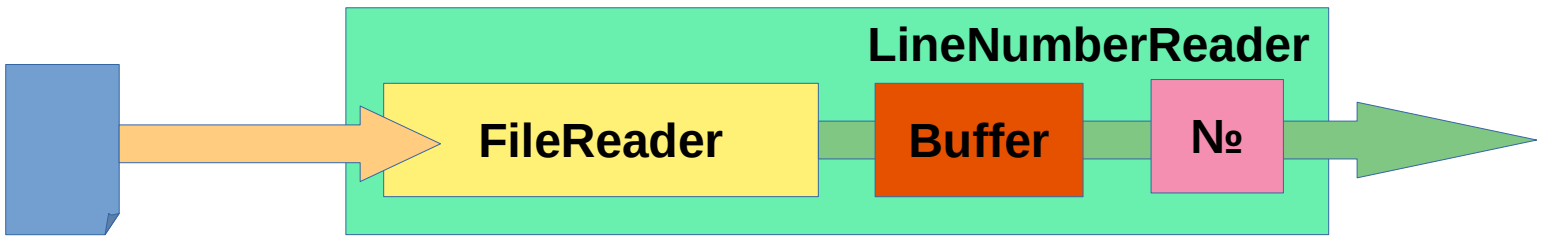
Потоки-фильтры - Buffered

- ☑ 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

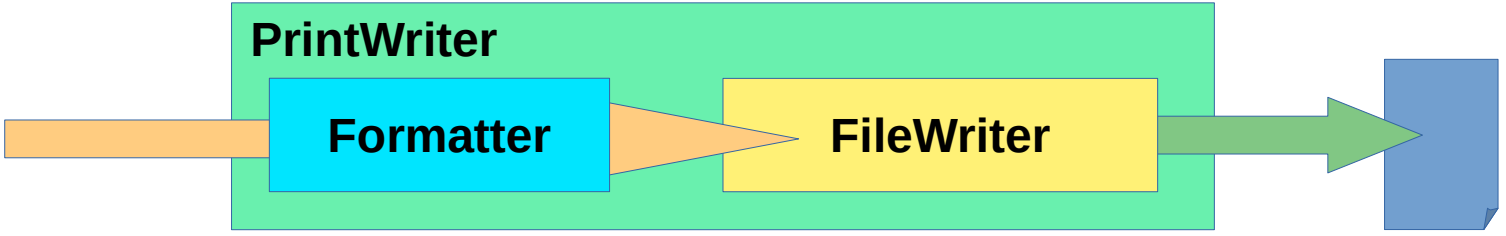
☑ Потоки-фильтры - байт ↔ символ

- ☑ 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*)

PrintStream, PrintWriter

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

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

π = +3,1415927
```



Java

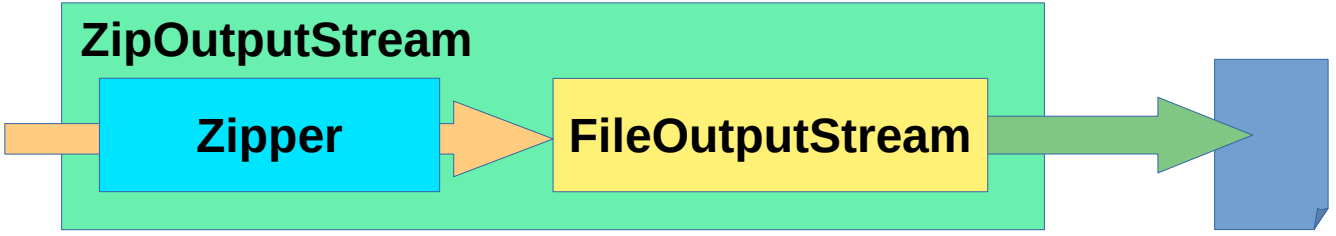
PrintStream, PrintWriter

- %[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

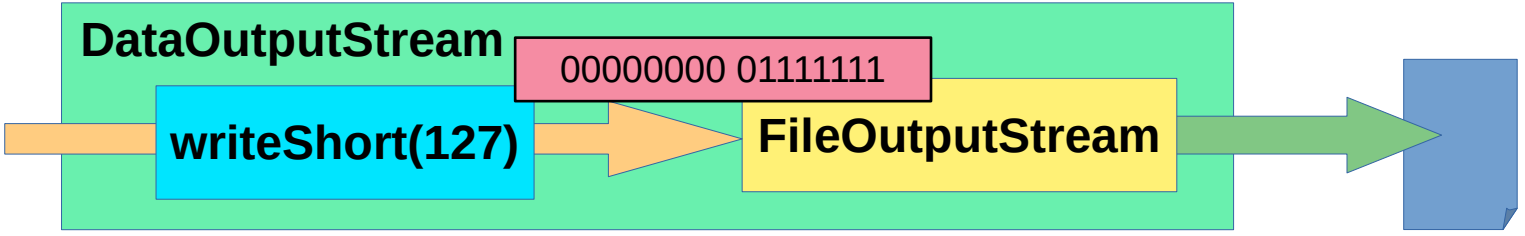
- ☑ потоки с компрессией
- ☑ InflaterInputStream
  - GzipInputStream
  - ZipInputStream
    - JarInputStream
- ☑ DeflaterOutputStream
  - GzipOutputStream
  - ZipOutputStream
    - JarOutputStream



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

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)

Java

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

ITMO re than a UNIVERSITY

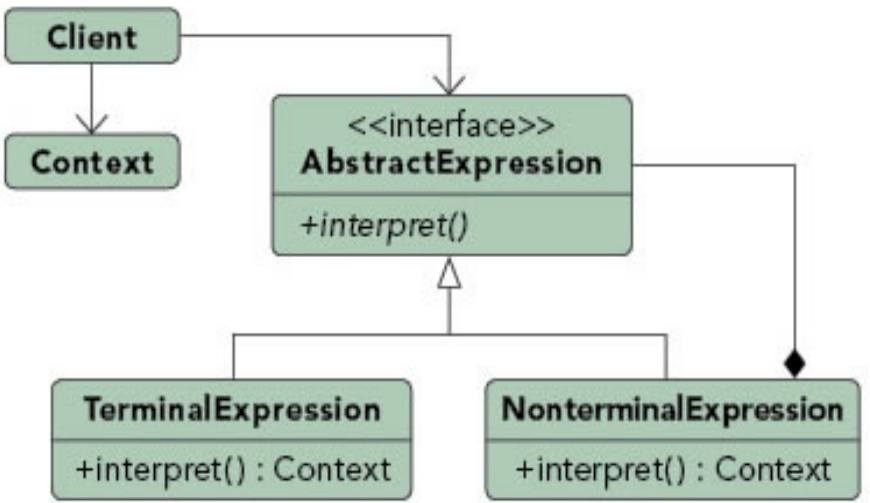
Класс java.util.Random

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

Java

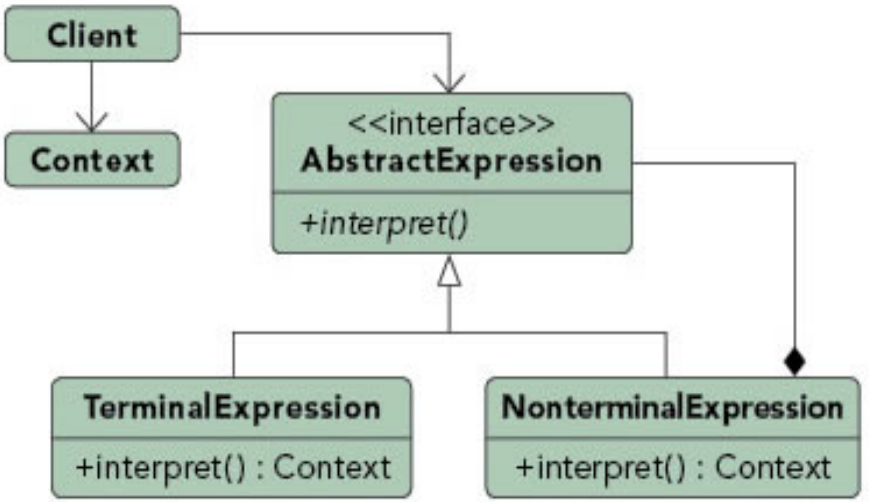
Interpreter

- ☑ Позволяет управлять поведением с помощью простого языка
  - Регулярные выражения
  - Форматирование строк



Interpreter

- ☑ Простота расширения и изменения языка
- ☑ Простота добавления новых способов
- ☑ Сложность сопровождения сложных грамматик



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

подстрока символов

```
in
```

начало и конец строки

```
^lo
```

```
ua$
```

```
lorem ipsum dolor sit
amet, consectetur
adipiscing elit, sed
do eiusmod tempor
incididunt ut labore
et dolore magna
aliqua
```

Java

Регулярные выражения

УНИВЕРСИТЕТ ИТМО

символьный класс

```
[bp]or
i[^dtns]
e[l-n]
```

```
lorem ipsum dolor sit
amet, consectetur
adipiscing elit, sed
do eiusmod tempor
incididunt ut labore
et dolore magna
aliqua
```

Java

Регулярные выражения

символьный класс -
сокращенное обозначение

```
\w\w\wt\W
```

\d	цифра	\D	не цифра
\w	буква	\W	не буква
\s	пробел	\S	не пробел

```
lorem ipsum dolor sit
amet, consectetur
adipiscing elit, sed
do eiusmod tempor
incididunt ut labore
et dolore magna
aliqua
```

Регулярные выражения

любой символ

```
...c
```

```
lorem ipsum dolor sit
amet, consectetur
adipiscing elit, sed
do eiusmod tempor_
incididunt ut labore
et dolore magna
aliqua
```

Регулярные выражения

альтернатива

```
s(i|ec)t
```

```
lorem ipsum dolor sit
amet, consectetur
adipiscing elit, sed
do eiusmod tempor
incididunt ut labore
et dolore magna
aliqua
```

Регулярные выражения

квантификаторы

```
\s.{6}\s
e.{0,2}i
s\S{0,1}m
```

```
lorem ipsum dolor sit
amet, consectetur
adipiscing elit, sed
do eiusmod tempor
incididunt ut labore
et dolore magna
aliqua
```

Java

Регулярные выражения

квантификаторы

`s\S?m`	`{0,1}`
`b.+d`	`{1,}`
`et[^o]*,`	`{0,}`

```
lorem ipsum dolor sit
amet, consectetur
adipiscing elit, sed
do eiusmod tempor
incididunt ut labore
et dolore magna
aliqua
```

Регулярные выражения

жадность

```
lor.*it
\si\w*?i
```

```
lorem ipsum dolor sit
amet, consectetur
adipiscing elit, sed
do eiusmod tempor
incididunt ut labore
et dolore magna
aliqua
```

Java

Регулярные выражения

группы

```
(\w+).*\1,
1 = it
```

```
lorem ipsum dolor sit
amet, consectetur
adipiscing elit, sed
do eiusmod tempor
incididunt ut labore
et dolore magna
aliqua
```

```
\w+@(\w+\.)+\w{2,}

user@se.itmo.ru
```

Регулярные выражения

УНИВЕРСИТЕТ ИТМО

группы

```
(\w)..\1(.).{1,6}\2
1 = i    2 = d
1 = t    2 = _
```

```
lorem ipsum dolor sit
amet, consectetur
adipiscing elit, sed
do eiusmod tempor
incididunt ut labore
et dolore magna
aliqua
```

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
```