Вопрос № 1

В каких классах пакета java.io можно вызвать метод `void flush()`?

Пишем конструкторы для всех вариантов ответа и добавляем метод. Всё, на что idea не ругается – правильные ответы. Важно, чтобы в конструкторах были написаны параметры, где это необходимо.

```
new ByteArrayOutputStream().flush();
new FileInputStream( name: "file").flush();
new CharArrayWriter().flush();
new BufferedWriter(new FileWriter( fileName: "file")).flush();
new FileOutputStream( name: "file").flush();
new BufferedReader(new FileReader( fileName: "file")).flush();
new CharArrayReader(new char[1]).flush();
new ByteArrayInputStream(new byte[1]).flush();
```

Ответ на вопрос №1 // Answer to question #1 *

Отметьте ВСЕ правильные ответы // Check ALL correct answers

- [x] ByteArrayOutputStream
- [ ] FileInputStream
- [x] CharArrayWriter
- [x] BufferedWriter
- [x] FileOutputStream
- [ ] BufferedReader
- [ ] CharArrayReader
- [ ] ByteArrayInputStream

Вопрос № 2

Какие значения элементов и в каком порядке будут находиться в коллекции после выполнения кода?

```
Set<Integer> set = new TreeSet<>();
set.add(5); set.add(4); set.add(3); set.add(2); set.add(4); set.add(1); set.remove(3);
```

Переписываем код в idea и выводим коллекцию.

Ответ на вопрос №2 // Answer to question #2 *

Введите элементы в нужном порядке через запятую без пробелов (например 1,2,3,4). Для Map использовать только значения, без ключей // Input elements in correct order separated by comma without spaces (for example 1,2,3,4). Use only values without keys for Map.

1,2,4,5

Вопрос № 3

Что напечатает фрагмент кода, если метод dupLast() удваивает последний символ?

```
Stream.of("january", "february", "march", "april", "may", "june")
.filter(s -> s.length() <= 7)
.map(s -> s.dupLast())
.skip(2)
.sorted()
.forEachOrdered(System.out::print);
```

Переписываем код в idea и заменяем метод на нужную комбинацию с использованием s.substring().

Ответ на вопрос №3 // Answer to question 3 *

Введите строку, которая будет выведена в результате выполнения кода // Input the string which will be printed after code execution

aprilljuneemayy

Вопрос № 4

Какой функциональный интерфейс соответствует лямбда-выражению?

```
(String s) -> s.hashCode()
```

Ответ на вопрос №4 // Answer to question #4 *

Выберите один правильный ответ // Choose one correct answer

- ○ Supplier<T>
- ○ Function<T, R>
- ○ UnaryOperator<T>
- ○ Predicate<T>
- ○ Consumer<T>
- ◉ ToIntFunction<T>

Сравниваем тип аргумента и результата по схеме:

```
Consumer<T>          -   (Type) argument --> void
Predicate<T>         -   (Type) argument --> (boolean) result
Supplier<T>          -    void --> (Type) result
BinaryOperator<T>    -   (Type) argument1, (Type) argument2 --> (Type) result
UnaryOperator<T>     -   (Type) argument --> (Type) result
ToIntFunction<T>     -   (Type) argument --> (int) result
Function<T,R>        -   (Type1) argument --> (Type2) result
```

Вопрос № 5

Какой класс используется для передачи потока байтов серверу при использовании протокола TCP?

Может попасться аналогичный вопрос про UDP. Важно обратить внимание на способ передачи данных (потоки / каналы для TCP, пакеты / каналы для UDP).

```
// Клиент TCP (потоки)
byte[] array = {0,1,2,3,4,5,6,7,8,9};
int port = 6789;
InetAddress host = InetAddress.getByName("localhost");
Socket socket = new Socket(host, port);
OutputStream outputStream = socket.getOutputStream();
InputStream inputStream = socket.getInputStream();

// Отправка потока на сервер
outputStream.write(array);

// Получение потока от сервера
int a = inputStream.read(array);
```

```
// Сервер TCP (потоки)
byte[] array = new byte[10];
int port = 6789;
InetAddress host = InetAddress.getByName("localhost");
Socket socket = new Socket(host, port);
ServerSocket serverSocket = new ServerSocket(port);
OutputStream outputStream = socket.getOutputStream();
InputStream inputStream = socket.getInputStream();

// Принятие запроса на подключение от клиента
serverSocket.accept();
// Получение потока от клиента
int a = inputStream.read(array);

// Отправка потока клиенту
outputStream.write(array);
```

```
// Клиент TCP NIO (каналы)
byte[] array = {0,1,2,3,4,5,6,7,8,9};
int port = 6789;
InetAddress host = InetAddress.getByName("localhost");
SocketAddress address = new InetSocketAddress(host,port);
SocketChannel socketChannel = SocketChannel.open();

// Отправка запроса на подключение к серверу
socketChannel.connect(address);
// Заполнение буфера для отправки
ByteBuffer buffer = ByteBuffer.wrap(array);
// Отправка буфера на сервер
socketChannel.write(buffer);
// Очистка буфера для приёма
buffer.clear();
// Приём буфера от сервера
socketChannel.read(buffer);
```

```
// Сервер TCP NIO (каналы)
byte[] array = new byte[10];
int port = 6789;
InetAddress host = InetAddress.getByName("localhost");
SocketAddress address = new InetSocketAddress(host, port);
ServerSocketChannel serverSocketChannel
        = ServerSocketChannel.open();
serverSocketChannel.bind(address);

// Принятие запроса на подключение от клиента
SocketChannel socketChannel = serverSocketChannel.accept();
// Заполнение буфера для отправки
ByteBuffer buffer = ByteBuffer.wrap(array);
// Приём буфера от клиента
socketChannel.read(buffer);
// Замена данных в буфере перед отправкой ответа
buffer.flip();
// Отправка буфера клиенту
socketChannel.write(buffer);
```

```java
// Клиент UDP (пакеты)
byte[] array = {0,1,2,3,4,5,6,7,8,9};
int port = 6789;
InetAddress host = InetAddress.getByName("localhost");
DatagramSocket datagramSocket = new DatagramSocket();

// Упаковка пакета для отправки
DatagramPacket datagramPacket
    = new DatagramPacket(array, array.length, host, port);
// Отправка пакета на сервер
datagramSocket.send(datagramPacket);
// Создание пакета для приёма
datagramPacket = new DatagramPacket(array, array.length);
// Приём пакета от сервера
datagramSocket.receive(datagramPacket);
```

```java
// Сервер UDP (пакеты)
byte[] array = new byte[10];
int port = 6789;
DatagramSocket datagramSocket = new DatagramSocket(port);
DatagramPacket datagramPacket
    = new DatagramPacket(array, array.length);

// Приём пакета от клиента
datagramSocket.receive(datagramPacket);

// Получение данных клиента для отправки ответа
InetAddress host = datagramPacket.getAddress();
port = datagramPacket.getPort();

// Упаковка пакета для отправки
datagramPacket
    = new DatagramPacket(array, array.length, host, port);
// Отправка пакета на клиент
datagramSocket.send(datagramPacket);
```

```java
// Клиент UDP NIO (каналы)
byte[] array = {0,1,2,3,4,5,6,7,8,9};
int port = 6789;
InetAddress host = InetAddress.getByName("localhost");
SocketAddress address = new InetSocketAddress(host, port);
DatagramChannel datagramChannel = DatagramChannel.open();

// Заполнение буфера для отправки
ByteBuffer buffer = ByteBuffer.wrap(array);
// Отправка буфера на сервер
datagramChannel.send(buffer, address);
// Очистка буфера для приёма
buffer.clear();
// Приём буфера от сервера
address = datagramChannel.receive(buffer);
```

```java
// Сервер UDP NIO (каналы)
byte[] array = new byte[10];
int port = 6789;
DatagramChannel datagramChannel = DatagramChannel.open();
SocketAddress address = new InetSocketAddress(port);
datagramChannel.bind(address);

// Создание буфера для приёма
ByteBuffer buffer = ByteBuffer.wrap(array);
// Приём буфера от клиента
address = datagramChannel.receive(buffer);

// Замена данных в буфере перед отправкой ответа
buffer.flip();
// Отправка буфера клиенту
datagramChannel.send(buffer, address);
```

Ответ на вопрос №5 или №11 // Answer to question #5 or #11 *

Введите название класса // Input class name

OutputStream

Вопрос № 6

Выберите необходимые инструкции для подключения к базе данных и сохранения результата запроса в переменной name и укажите их правильный порядок. Аргументы методов значения не имеют.

```
1. if (rs.next()) String name = rs.getString(...);
2. DriverManager dm = new DriverManager(...);
3. Connection conn = dm.connect(...);
4. Statement st = conn.executeStatement(...);
5. Connection conn = DriverManager.getConnection(...);
6. ResultSet rs = st.getResultSet(...);
7. ResultSet rs = st.executeQuery(...);
8. if (rs.next()) String name = rs.nextString(...);
9. Statement st = conn.prepareStatement(...);
```

Ответ на вопрос №6 // Answer to question #6 *

Введите номера только корректных инструкций в порядке их выполнения через запятую без пробелов (например 1,2,3,4) // Input only indices of correct statements in order of their execution separated by comma without spaces (for example 1,2,3,4)

5,9,7,1

Вопрос одинаковый для всех вариантов, меняются только цифры. Нужный порядок:

```
Connection conn = DriverManager.getConnection(...);
Statement st = conn.prepareStatement(...);
ResultSet rs = st.executeQuery(...);
if(rs.next) String name = rs.getString(...);
```

Вопрос № 7

Из каких состояний поток может перейти из состояния **Runnable(Ready)** ?

Вопрос может быть некорректно сформулирован (как в примере). Как правильно интерпретировать – без понятия. Схема переходов:

```
new -> runnable
runnable -> running
running -> waiting, blocked, terminated, runnable
waiting -> runnable
blocked -> runnable
terminated -> никуда
```

Ответ на вопрос №7 // Answer to question #7 *

Отметьте ВСЕ правильные ответы. // Check ALL correct answers.

- [x] New
- [x] Runnable (Running)
- [x] Blocked
- [x] Waiting
- [ ] Sleeping
- [ ] Terminated
- [ ] Runnable (Ready)

Вопрос № 8

Какая строка должна быть в файле `hello de DE.properties`, чтобы в результате выполнения кода было напечатано `'hello weld'` ?

```
Locale locale = new Locale("de_DE");
ResourceBundle bundle = ResourceBundle.getBundle("hello", locale);
String hello = bundle.getString("hello");
String world = bundle.getString("world");
System.out.println(hello + " " + world);
```

Содержимое файла `hello.properties`:

```
hello = hello
world = world
bundle = bundle
```

Ответ на вопрос №8 // Answer to question #8 *

Какую строку нужно добавить в файл свойств? // Which line should be added to the properties file?

world = weld

Вместо ненужного слова подставить нужное. Ответ в формате: старое слово = новое слово

Вопрос № 9

Впишите недостающий элемент в код обработки события.

```
JTextField c = new JTextField("The Empire Strikes Back");
c._____________________(new ActionListener() {
  public void actionPerformed(ActionEvent ev) {
    this.setBackground(Color.PINK);
  }
});
```

Ответ на вопрос №9 // Answer to question #9 *

Введите пропущенный элемент // Input the missing element

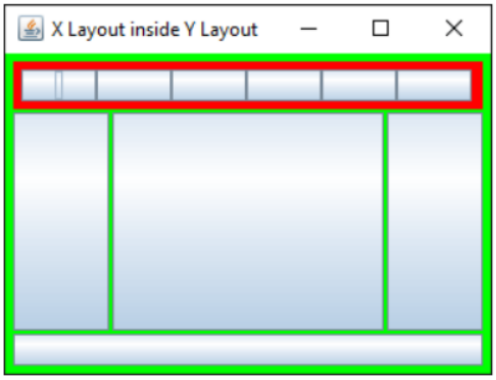
addActionListener

Переписать первую строчку в idea, на месте пропуска чаще всего находится метод, начинающийся с `add`... В появившемся после ввода `c.add` списке найти наиболее логичный вариант и проверить, что idea на него не ругается.

```
JTextField c = new JTextField("The Empires Strikes Back");
c.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ev) {
        c.setBackground(Color.PINK);
    }
});
```

Вопрос № 10

10 Какой менеджер компоновки установлен у панели с зеленым фоном?



Ответ на вопрос №10 // Answer to question #10 *

Выберите менеджер компоновки Swing или панель JavaFX с компоновкой, соответствующей панели с зеленым фоном на картинке // Choose a Swing layout manager or JavaFX pane with layout, which is matching to the panel with green background from the picture

- ○ BoxLayout (X_AXIS) / HBox
- ○ SpringLayout / AnchorPane
- ○ CardLayout / StackPane
- ◉ BorderLayout / BorderPane
- ○ TableLayout / TablePane
- ○ BoxLayout (Y_AXIS) / VBox
- ○ GridLayout / TilePane

Краткое описание лэйаутов:

```
FlowLayout      -   Дефолтный для JPanel. Последовательное расположение построчно (слева направо, сверху вниз)
BoxLayout       -   Последовательное расположение вертикально (Y_AXIS) или горизонтально (X_AXIS)
BorderLayout    -   Расположение по границам окна (NORTH, SOUTH, WEST, EAST, CENTER)
GridLayout      -   Табличное расположение (ячейки одного размера)
GridBagLayout   -   Табличное расположение (ячейки произвольного размера)
TableLayout     -   Табличное расположение (произвольные размеры строк и столбцов)
CardLayout      -   Расположение для вкладок (выбор отображения элемента среди занимающих одно место)
SpringLayout    -   Расположение по расстоянию между парами границ элементов
```