www.performance-lab.ru

Замыкания

```
public class LambdasClosures {
    public Function<Integer, Integer> make_fun() {
    // вне области видимости возвращенной функции:
    int n = 0;
    return arg -> {
        System.out.print(n + " " + arg + ": ");
        arg += 1;
        // n += arg; // ошибка
        return n + arg;
    };
}}
```

Ввод/вывод

Ввод и вывод в Java представлен абстрактным понятиям «Поток» (Stream) и базируется на двух абстрактных классах InputStream и OutputStream.
Для работы с файлами существуют классы FileInputStream, FileOutputStream.
FileReader и FileWriter чуть более удобные классы для работы с файлами.
BufferedReader, BufferedWriter - классы-обертки над символьными потоками.
BufferedInputStream, BufferedOutputStream – обертки над бинарными потоками.
Scanner – удобный класс для разбора строк.
System.in, System.out, System.err – стандартные потоки ввода/вывода.
Все потоки должны быть закрыты вручную методом close.
Все вышеуказанные классы реализуют интерфейс Autoclosable.

Ввод/вывод

```
FileInputStream in;
try {
    in = new FileInputStream("a.txt");
    int c;
    while ((c = in.read()) != -1) {
        System.out.write(c);
    }
}
catch (IOException e) { e.printStackTrace(); }
finally {
   try { in.close(); }
   catch (IOException ignored) {}
}
```

Ввод/вывод

Try с ресурсами

Такая форма оператора try применима к объектам, реализующим интерфейс java.lang.AutoCloseable.

```
try (FileInputStream in = new FileInputStream("a.txt")) {
    int c;
    while ((c = in.read()) != -1) {
      System.out.write(c);
    }
}
catch (IOException e) { e.printStackTrace(); }
```

Ввод/вывод

Try с ресурсами

При этом, если в блоке *try-with-resources* произошли исключения в блоке try и при попытке закрыть ресурсы, то исключеие из блока *try* будет основным. Все остальные выброшенные исключения будут добавлены в качестве подавленных. Их можно получить методом

```
public final Throwable[] getSuppressed()
```

Сериализация

Чтобы иметь возможность сериализации, класс должен быть помечен маркерным интерфейсом Serializable.

По умолчанию сохраняется все состояние объекта.

Если какое-то поле сохранять не нужно, его можно пометить ключевым словом transient.

Если внутри объекта используются поля других объектов, они так же должны быть помечены Serializable.

Сериализация

```
import java.io.Serializable;

class TestSerial implements Serializable {
  public byte version = 100;
  public byte count = 0;

  public static void main(String args[]) throws IOException {
    FileOutputStream fos = new FileOutputStream("temp.out");
    ObjectOutputStream oos = new ObjectOutputStream(fos);
    TestSerial ts = new TestSerial();
    oos.writeObject(ts);
    oos.flush();
    oos.close();
}}
```

Сериализация

```
public static void main(String args[]) throws IOException {
  FileInputStream fis = new FileInputStream("temp.out");
  ObjectInputStream oin = new ObjectInputStream(fis);
  TestSerial ts = (TestSerial) oin.readObject();
  System.out.println("version=" + ts.version);
}
```

PerformanceLab

Обеспечение производительности ИТ

Спасибо за внимание!