Лекция 4.3: Ввод/вывод / Serialization API.

Intro

1. Статические поля не сериализуются
2. Конструктор не вызывается
3. transient
4. runtime определение – кидать исключение или нет

Критика сериализации:
1. Формат – бинарный.
1. Формат – не компактный.
1. Формат – не кроссплатформенный.
5. Ограничивает изменения классов в будущем (нельзя изменить имя класса)

1. Читать Блоха
2. Читать Елизарова,

private void writeObject(java.io.ObjectOutputStream out)
throws IOException {
// write ‘this’ to ‘out’…

private void readObject(java.io.ObjectInputStream in)
throws IOException, ClassNotFoundException {
// populate the fields of ‘this’ from the data in ‘in’…
}
}

Тесты по всей лекции

Тест, состоящий из случайных вопросов тестов этой лекции

Some issues occured with quiz randomizer :(

Видео

??? Набор декабрь 2013
??? Набор октябрь 2013
??? Набор июль 2013
??? Набор апрель 2013
??? Набор февраль 2013
??? Набор январь 2013
??? Набор октябрь 2012

Сложное НЕ мое видео
Роман Елизаров, Факты и заблуждения о Java-сериализации

Набор февраль 2013 (сер + клонирование)
Набор октябрь 2012

Лабораторные

io.serialization.active_object
Task – интерфейс характеризующий некоторую многоэтапную работу.

```java
public interface Task {
    public Task next();
    public long sleepTime();
}
```

Ожидаемое использование Task-a

```java
while (task != null) {
     Thread.sleep(task.sleepTime());
     task = task.next();
}
```

HelloTask – специальный Task, который counter раз шлет Вам привет.

```java
import java.io.*;

/**
 * Serializable WITH writeObject/readObject
 */
public class HelloTask implements Task, Serializable {
    private String msg;
    private int counter;
    private long sleepTime;

    public HelloTask(String msg, int counter, long sleepTime) {
        this.msg = msg;
        this.counter = counter;
        this.sleepTime = sleepTime;
    }

    @Override
    public Task next() {
        System.out.println(msg + ":" + counter);
        counter--;
        return (counter == 0) ? null : this;
    }

    @Override
    public long sleepTime() {
        return sleepTime;
    }

    public String getMsg() {
        return msg;
    }

    public int getCounter() {
        return counter;
    }

    public long getSleepTime() {
        return sleepTime;
    }

    private void writeObject(ObjectOutputStream out) throws IOException {
        out.writeUTF(msg);
        out.writeInt(counter);
        out.writeLong(sleepTime);
    }

    private void readObject(ObjectInputStream in) throws IOException, ClassNotFound
        this.msg = in.readUTF();
        this.counter = in.readInt();
        this.sleepTime = in.readLong();
    }
}
```

TaskThread – поток, выполняющий Task. Поток допускает приостановку и повторный запуск.

```java
public class TaskThread extends Thread {
    private Task task;
    private volatile boolean pause;

    public TaskThread(Task task) {
        this.task = task;
    }

    public void pauseOn() {
        this.pause = true;
        this.interrupt();
    }

    public void pauseOff() {
        this.pause = false;
    }

    @Override
    public void run() {
        while (task != null) {
            try {
                Thread.sleep(task.sleepTime()); //wait for next step
            } catch (InterruptedException e) {
                if (pause) {
                    while (pause); // wait for 'pauseOff'
                } else {
                    return;
                }
            }
            task = task.next(); // make next step
        }
    }

    public Task getTask() {
        return task;
    }
}
```

ActiveObject – “активный объект”, сочетающий в себе И поток выполнения И выполняемую задачу. Можно сказать, что это объект, который вызывает методы сам у себя.

```java
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;


public class ActiveObject implements Serializable {
    private transient TaskThread thread;

    public ActiveObject(Task task) {
        this.thread = new TaskThread(task);
        this.thread.start();
    }

    public void pauseStart() {
        thread.pauseOn();
    }

    public void pauseStop() {
        thread.pauseOff();
    }

    private void writeObject(ObjectOutputStream out) throws IOException {
        // stop my activity
        thread.pauseOn();
        // serialize internal task
        Task task = thread.getTask();
        out.writeObject(task);
        // restart my activity
        thread.pauseOff();
    }

    private void readObject(ObjectInputStream in) throws IOException, ClassNotFound
        // deserialize internal task
        Task task = (Task) in.readObject();
        // start new activity
        this.thread = new TaskThread(task);
        this.thread.start();
    }
}
```

ActiveObject_Demo – демонстрация работы с активным объектом

```java
import java.io.*;

public class ActiveObject_Demo {
    public static void main(String[] args) throws Exception {

        ActiveObject oldActiveTask = new ActiveObject(new HelloTask("Hello", 20, 50
        Thread.sleep(5000);

        byte[] rawData = writeActiveTask(oldActiveTask);
        System.out.println("rawData.length = " + rawData.length);
        System.out.println(new String(rawData));
        ActiveObject newActiveTask_0 = readActiveTask(rawData);
        ActiveObject newActiveTask_1 = readActiveTask(rawData);
        ActiveObject newActiveTask_2 = readActiveTask(rawData);
    }

    private static byte[] writeActiveTask(ActiveObject activeTask) throws IOExcepti
        ByteArrayOutputStream buff = new ByteArrayOutputStream();
        try (ObjectOutput out = new ObjectOutputStream(buff)) {
            out.writeObject(activeTask);
            out.flush();
            return buff.toByteArray();
        }
    }

    private static ActiveObject readActiveTask(byte[] rawData) throws IOException,
        try (ObjectInputStream src = new ObjectInputStream(new ByteArrayInputStream
            return (ActiveObject) src.readObject();
        }
    }
}

>> Hello:20
>> Hello:19
>> Hello:18
>> Hello:17
>> Hello:16
>> Hello:15
>> Hello:14
>> Hello:13
>> Hello:12
>> Hello:11
>> Hello:10
>> rawData.length = 235
...
>> Hello:9
>> Hello:10
>> Hello:10
>> Hello:10
>> Hello:8
>> Hello:9
>> Hello:9
>> Hello:9
>> Hello:7
>> Hello:8
>> Hello:8
>> Hello:8
>> Hello:6
>> Hello:7
>> Hello:7
>> Hello:7
...
>> Hello:3
>> Hello:1
>> Hello:2
>> Hello:2
>> Hello:2
>> Hello:1
>> Hello:1
>> Hello:1
```

Задание:
Переписать методы writeObject()/readObject() класса ActiveTask следующим образом

```java
private void writeObject(ObjectOutputStream out) throws IOException {
    thread.pauseOn();
    Task task = thread.getTask();
    if (task instanceof HelloTask) {
        HelloTask hello = (HelloTask) task;
        // сохранить поля HelloTask
    } else {
        out.writeObject(task);
    }
    thread.pauseOff();
}
```

```java
private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundExce
    boolean isHelloTask = in.readBoolean();
    if (isHelloTask) {
        // восстановить поля HelloTask
    } else {
        this.thread = new TaskThread((Task) in.readObject());
    }
    this.thread.start();
}
```

т.е. в случае если полем типа Task является конкретно HelloTask использовать не автоматическую сериализацию, а вручную сохранить поля пользуять тем, что ObjectOutputStream/ObjectInputStream являются также DataOutput/DataInput.
Цель заключается в том, что бы в случае, если ActiveObject работает с HelloTask сократить сериализованный размер с 235 до 100-130 байт.
Дополнительное задание: удастся ли сократить хранимый размер, если сделать?

```java
public class ActiveObject implements java.io.Externalizable {
    ....
}
```

Примечание:
на текущий момент к этой лабораторной нет автоматической проверки.

Литература

Хорстманн, Корнелл. “Java 2. Том I. Основы”. Издание 7
….Глава 12. Потоки и файлы, стр 778-804