СБЕРБАНК ТЕХНОЛОГИИ

Java Serialization

УЗНАЕМ

СБЕРБАНК ТЕХНОЛОГИИ

- Что такое Serialization в Java
- Как работает сериализация
- Что такое версия класса
- Как сериализовать объект и восстановить его состояние

ЗАЧЕМ?

СБЕРБАНК ТЕХНОЛОГИИ

- Осуществление передачи данных между процессами
- Сохранение пользовательских настроек / данных приложения от запуска к запуску
- …

Неужели каждый раз нужно заново разрабатывать механизм сохранения и считывания состояния объекта?

ЧТО ТАКОЕ SERIALIZATION?

СБЕРБАНК ТЕХНОЛОГИИ

- *Сериализация* – это процесс сохранения состояния объекта в последовательность байт;

- *Десериализация* – это процесс восстановления объекта из этих байт.

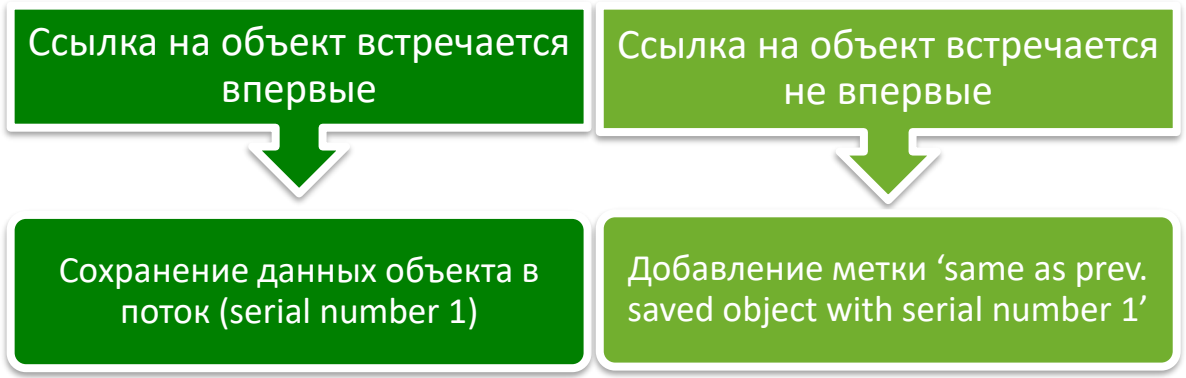
Кто ответит откуда взялось такое название?

СХЕМА СЕРИАЛИЗАЦИИ ОБЪЕКТА ССЫЛОЧНОГО ТИПА

СБЕРБАНК ТЕХНОЛОГИИ

Присвоение объекту серийного номера (serial number)

Соотнесение serial number и каждой ссылки на данный объект



КРАТКО: JAVA SERIALIZATION API

СБЕРБАНК ТЕХНОЛОГИИ

- **interface** Serializable { }
- **interface** Externalizable **extends** java.io.Serializable {

  void writeExternal(ObjectOutput out) **throws** IOException;

  void readExternal(ObjectInput in) **throws** IOException, ClassNotFoundException;

  }
- **class** ObjectOutputStream **extends** OutputStream …
- **class** ObjectInputStream **extends** InputStream …
- **transient** keyword
- чёрная магия

Для встроенного механизма сериализации необходимо чтобы

1. Класс реализовывал интерфейс java.io.Serializable;
2. Были определены поля, участвующие в сериализации
   - Стандартно это все поля не помеченные ключевым словом transient
   - Можно явно указать список полей для сериализации

```
private static final ObjectStreamField[] serialPersistentFields =
                                { new ObjectStreamField("name", String.class) };
```

3. Был доступен конструктор без аргументов первого не сериализуемого родительского класса

ПРИМЕР СЕРИАЛИЗУЕМОГО ОБЪЕКТА

СБЕРБАНК ТЕХНОЛОГИИ

```
class SerializableObject implements Serializable {
    private Student[] students = {new Student(”Name")};

    // ...
}

class Student implements Serializable {
    private String name;

    Student(String name) {
        this.name = name;
    }

    // ...
}
```

КАК СЕРИАЛИЗОВАТЬ ДАННЫЕ В ФАЙЛ?

СБЕРБАНК ТЕХНОЛОГИИ

```
void serialize(String filename, Student student) throws IOException {
    try (FileOutputStream fos = new FileOutputStream(filename);
         ObjectOutputStream out = new ObjectOutputStream(fos))    {
        out.writeObject(student);
    } }

Student deserialize(String filename) throws IOException, ClassNotFoundException {
    try (FileInputStream fis = new FileInputStream(filename);
         ObjectInputStream in = new ObjectInputStream(fis))    {
        return (Student) in.readObject();
    } }
```

АЛГОРИТМ СЕРИАЛИЗАЦИИ

СБЕРБАНК ТЕХНОЛОГИИ

1. запись метаданных о классе ассоциированном с объектом
2. рекурсивная запись описания **суперклассов**, до тех пор пока не будет достигнут *java.lang.Object*
3. запись фактических данных, ассоциированных с экземпляром, начинается **с самого верхнего суперкласса**

БАЙТОВОЕ ПРЕДСТАВЛЕНИЕ

СБЕРБАНК ТЕХНОЛОГИИ

```
0000000 ac ed 00 05 73 72 00 0e 72 75 2e 73 62 74 2e 53
0000010 74 75 64 65 6e 74 a9 73 e0 6c ef d1 4b ba 02 00
0000020 01 4c 00 04 6e 61 6d 65 74 00 12 4c 6a 61 76 61
0000030 2f 6c 61 6e 67 2f 53 74 72 69 6e 67 3b 78 70 74
0000040 00 04 4e 61 6d 65
```

AC ED: STREAM_MAGIC. Говорит о том, что используется протокол сериазизации.
00 05: STREAM_VERSION. Версия сериализации.
73: TC_OBJECT. Обозначение нового объекта. 72: TC_CLASSDESC. Обозначение нового класса.
00 0e: Длина имени класса. 72 75 2e 73 62 74 2e 53 74 75 64 65 6e 74: ru.sbt.Student.
a9 73 e0 6c ef d1 4b ba: serialVersionUID, идентификатор класса.
02: Различные флаги. Этот специфический флаг говорит о том, что объект сериализуемый.
00 01: Число полей в классе. 0x4c: Идентификатор типа – Строка 00 04 – длина имени поля
6e 61 6d 65: name, 74: Новая строка, 00 12: длина строки, Ljava/lang/String;
0x78: TC_ENDBLOCKDATA, конец опционального блока данных для объекта.
0x70: TC_NULL, обозначает то что больше нет суперклассов. мы достигли верха иерархии классов.
74 00 04 4e 61 6d 65: 4-х символьная строка «Name»

TRANSIENT ДАННЫЕ

СБЕРБАНК ТЕХНОЛОГИИ

```
class User implements Serializable {
    String login;
    transient String password;

    User(String login, String password) {
        this.login = login;
        this.password = password;
    }
    // ...
}
```

```
ObjectInputStream objectInputStream = ...

user = (User) objectInputStream.readObject();

System.out.println(user.getPassword()); // ?
```

```
ObjectOutputStream out = new ObjectOutputStream(...);
Student obj = new Student(“Ivan”);
out.writeObject(obj); // сохраняет объект с состоянием name=Ivan
obj.setName(“Petr”);
out.writeObject(obj); // не сохраняет новое состояние объекта
```

Как избежать:

- каждый раз после вызова метода записи закрывать поток;
- вызвать метод *ObjectOutputStream.reset()*, котрый сообщает что нужно освободить кэш от ссылок.

РАСШИРЕНИЕ ПРОТОКОЛА СЕРИАЛИЗАЦИИ

СБЕРБАНК ТЕХНОЛОГИИ

```
private void writeObject(ObjectOutputStream out) throws IOException {
    out.defaultWriteObject();
    out.writeLong(Calendar.getInstance().getTimeInMillis());
}

private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException {
    in.defaultReadObject();
    long writeTime = in.readLong();
    System.out.println(writeTime);
}
```

ПОТОМОК ПЕРЕСТАЛ БЫТЬ СЕРИАЛИЗУЕМЫМ

СБЕРБАНК ТЕХНОЛОГИИ

```
class SerializableParent implements Serializable {
    String department;
    // ...
}

class NotSerializableChild extends SerializableParent {
    String securedData;

    // ...
}
```

ПОТОМОК НЕ ДОЛЖЕН БЫТЬ СЕРИАЛИЗУЕМЫМ

СБЕРБАНК ТЕХНОЛОГИИ

```
class SerializableParent implements Serializable {
    String department;
    // ...
}

class NotSerializableChild extends SerializableParent {
    String securedData;

    private void writeObject(ObjectOutputStream stream) throws IOException {
        throw new NotSerializableException();
    }
    // ...
}
```

ВОССТАНОВЛЕНИЕ ОБЪЕКТА

СБЕРБАНК ТЕХНОЛОГИИ

? Сможем ли мы восстановить объект, если после его сериализации:

- изменится имя поля сериализованного уже в файл класса
- поменялся только тип
- всего лишь изменился модификатор доступа к полю
- Изменился как-либо метод
- JVM, сериализовавшая объект, отличается от считывающей его JVM

ВЕРСИЯ КЛАССА

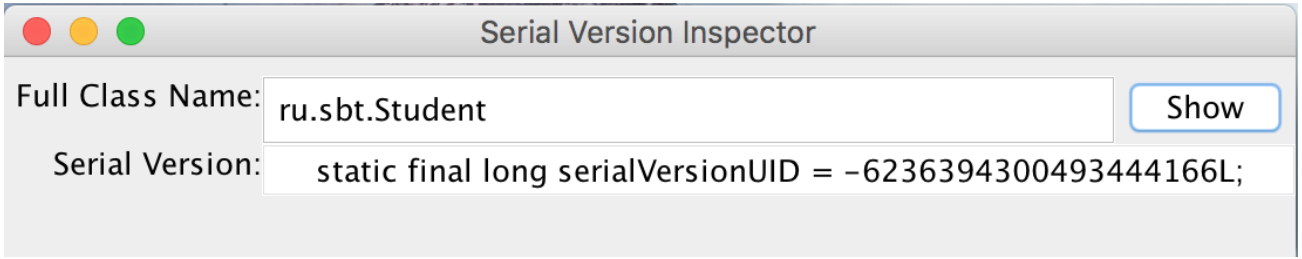
СБЕРБАНК ТЕХНОЛОГИИ

private static final long *serialVersionUID* = 7735307134757953751;

Если явно не указан в классе то он вычисляется на основе хэш SHA-1 используя

- Имя класса
- Имена интерфейсов
- Имена всех методов и полей
- … см. спецификацию

Может быть рассчитан с помощью утилиты из JDK – **serialver**



СОВМЕСТИМЫЕ ИЗМЕНЕНИЯ

СБЕРБАНК ТЕХНОЛОГИИ

Это изменения класса после сериализации его предыдущей версии, гарантирующие совместимость данных после десериализации в обновленный класс.

- Добавление новых полей
- Добавление и удаление классов в иерархию класса
- Добавление и удаление методов writeObject/readObject
- Реализация java.io.Serializable - аналогично добавлению нового класса
- Изменение модификаторов доступа к полям
- Изменение поля static -> nonstatic или transient -> nontransient – аналогично добавлению нового поля

СОБСТВЕННЫЙ ПРОТОКОЛ

СБЕРБАНК ТЕХНОЛОГИИ

```
package java.io;

interface Externalizable extends Serializable {
    public void writeExternal(ObjectOutput out)
            throws IOException;

    public void readExternal(ObjectInput in)
            throws IOException, java.lang.ClassNotFoundException;
}
```

EXTERNALIZABLE INTERFACE

СБЕРБАНК ТЕХНОЛОГИИ

Externalizable объект

- Реализовывает интерфейс java.io.Externalizable
- Имеет конструктор без аргументов
- Отвечает за сериализацию данных своего класса и всех его родительских классов

EXTERNALIZABLE VS SERIALIZABLE

СБЕРБАНК ТЕХНОЛОГИИ

Externalizable	Serializable
Свои правила и свой механизм сериализации объектов	Предоставляет механизм сериализации и стандартный протокол «из коробки»
Наследуется от Serializable	Маркер-интерфейс
Ответсвенность сохранения состояния родительского класса на реализующем классе	Поддерживается сериализации состояния всех сериализуемых родительских классов
writeExternal & readExternal заменяют магические writeObject & readObject	Расширение стандартного механизма с помощью writeObject\readObject
Объект создается публичным конструктором без аргументов	Рефликсивное наполнение без вызова конструктора
Требование иметь конструктор	Конструткор без аргументов не обязателен

SERIALIZATION EXCEPTIONS

СБЕРБАНК ТЕХНОЛОГИИ

- **ObjectStreamException**
  - Суперкласс всех ошибок сериализации
- **InvalidClassException**
  - Не совпадает версия класса
  - Примитивные типы хранят недопустимые значения
  - Externalizable не имеет публичного конструктора без параметров
  - Serializable класс имеет несериализуемого предка без конструктора
- **NotSerializableException**
  - Бросается в readObject\writeObject для остановки механизма сериализации
- **InvalidObjectException**
  - Бросается для обозначения что считанный объект в несогласованном состоянии
- **ClassNotFoundException**
  - Класс не найден

ШИФРОВАНИЕ И ПОДПИСЬ СЕРИАЛИЗУЕМЫХ ОБЪЕКТОВ

СБЕРБАНК ТЕХНОЛОГИИ

Классы обертки:

- **java.security.SignedObject**
  - подписывает объект цифровой подписью, защищая целостность и аутентичность данных
  - нужен симметричный ключ
- **javax.crypto.SealedObject**
  - защищает криптографическим алгоритмом, скрывая данные
  - нужен симметричный ключ

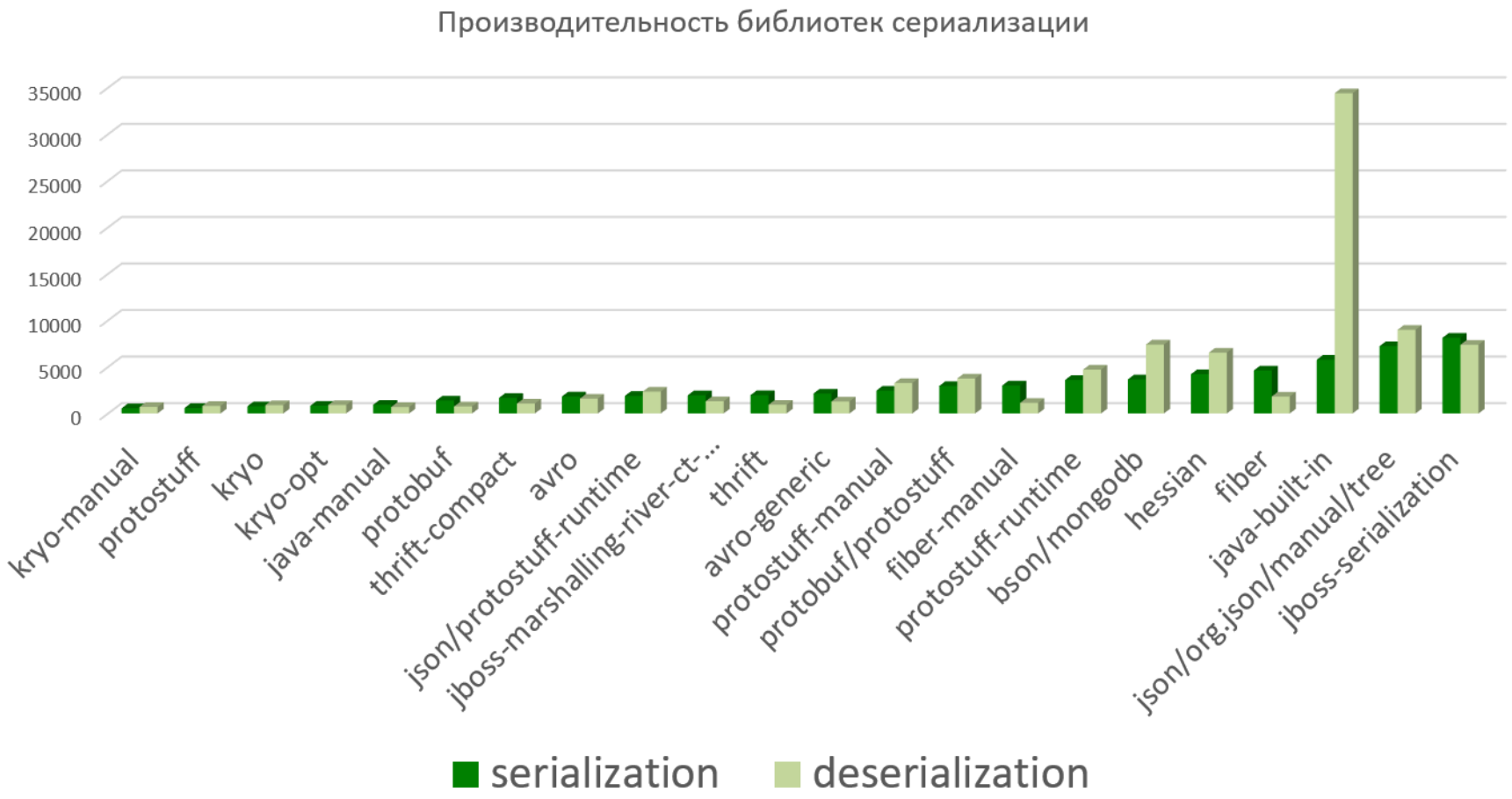
БИБЛИОТЕКИ ЭФФЕКТИВНОЙ СЕРИАЛИЗАЦИИ

СБЕРБАНК ТЕХНОЛОГИИ

- Protocol Buffer (protobuf)
  - protostuff (the **stuff** that leverages google's **proto**buf)
  - colfer (Proto**col** Buf**fer**)
- Kryo Serializer
- Apache Avro
- fst (fast-serialization)
- Fiber
- **и десятки других…**

БИБЛИОТЕКИ ЭФФЕКТИВНОЙ СЕРИАЛИЗАЦИИ

СБЕРБАНК ТЕХНОЛОГИИ



БИБЛИОТЕКИ ЭФФЕКТИВНОЙ СЕРИАЛИЗАЦИИ

СБЕРБАНК ТЕХНОЛОГИИ

ВЫВОДЫ

СБЕРБАНК ТЕХНОЛОГИИ

- Java Seriaization – простой механизм сериализации данных
- Встроенная поддержка безопасной типизации объектов и их свойств в сериализованном виде
- Обеспечена возможность расширения протокола сериализации для поддержки маршалинга и демаршалинга удаленных объектов при RPC вызовах
- Обеспечена возможность расширения протокола сериализации для поддержки персистентности объектов
- Особенности сериализации добавляются только в классах, хранящих эти особенности
- Объекты могут сами определять свой протокол сериализации

МАТЕРИАЛЫ

СБЕРБАНК ТЕХНОЛОГИИ

- https://m.habrahabr.ru/post/60317/
- http://www.javaworld.com/article/2076120/java-se/flatten-your-objects.html
- http://www.javaworld.com/article/2072752/the-java-serialization-algorithm-revealed.html
- https://docs.oracle.com/javase/8/docs/technotes/guides/serialization/relnotes.html
- http://javapapers.com/core-java/externalizable-vs-serializable/
- http://docs.oracle.com/javase/6/docs/platform/serialization/spec/serialTOC.html