Прекрасный мир Java

Часть 12.

Сериализация

Процесс преобразования объекта класса в бинарный формат, пригодный для размещения в хранилище, называется **сериализацией**.

Обратный процесс восстановления объекта из бинарного формата называется **десериализацией**.

Способность к сериализации – это свойство конкретного класса.

Интерфейс **Serializable**

Минимальные требования к сериализуемому классу:

- реализация интерфейса Serializable
- наличие конструктора (возможно, закрытого) без параметров.

Процесс сериализации в JVM

1. При вызове метода ObjectOutputStream.writeObject делается проверка, что параметр метода реализует интерфейс Serializable. Если проверка не проходит, то генерируется исключение.

2. JVM записывает имя и версию класса (версия получается из статического поля serialVersionUID, если таковое присутствует в классе; иначе версия строится автоматически по сигнатуре класса).

3. Если в сериализуемом классе присутствует метод writeReplace(), то он вызывается, и возвращенный им результат используется для сериализации.

4. JVM последовательно просматривает поля объекта. Поля примитивных типов записываются примерно тем же способом, как в рамках класса DataOutputStream. Поля объектных типов подвергаются записи путем рекурсивного вызова writeObject. Массивы обрабатываются специальным образом.

5. Получившийся в результате «пакет» информации передается в нижележащий бинарный поток.

Процесс десериализации в JVM

1. Считывается пакет информации из нижележащего бинарного потока.

2. Из пакета выделяется имя класса, экземпляр которого следует реконструировать. Если описатель этого класса недоступен, генерируется исключение ClassNotFoundException (в этом случае, очевидно, десериализация невозможна).

3. JVM создает экземпляр класса путем особого (в обход обычных правил доступа к методам) вызова конструктора без параметров.

4. Считывается версия класса и делается проверка на совпадение с версией класса, созданного в п. 3.

5. JVM последовательно просматривает поля объекта. Поля примитивных типов восстанавливаются путем чтения информации по аналогии с тем, как это делает DataInputStream. Поля объектных типов восстанавливаются

6. Если в классе реализован метод readResolve(), то производится его вызов. Возвращенный им результат становится результатом десериализации.

Сериализация позволяет полностью сохранить состояние объекта.

Таким образом, сериализацию обычно применяют в следующих ситуациях:
1. Сохранении наборов данных во внешних хранилищах (фактически, это аналогично тому, как сохраняются документы в офисных программах).
2. Для организации межпрограммного взаимодействия (в этом случае сериализованные классы передаются по сети между разными программами или даже разными системами).

Программист имеет возможность явного управления процессом сериализации путем определения в своем классе следующих двух методов:

```
private void writeObject (ObjectOutputStream
out) throws IOException

private void readObject (ObjectInputStream in)
throws IOException, ClassNotFoundException
```

Метод writeObject должен обеспечить реализацию логики сериализации, а метод readObject — десериализации.