Кравченко Даниил Андреевич
студент
ФГБОУ ВО «Государственный университет
морского и речного флота
им. адмирала С.О. Макарова»
г. Санкт-Петербург

СЕРИАЛИЗАЦИЯ В ООП JAVA

Аннотация*: статья посвящена понятию сериализации и связанным с ней процессам в объектно-ориентированном языке программирования Java. Приведены и описаны такие понятия как, сериализация и ее возможности, персистентность, потоки. Автором рассмотрены классы и интерфейсы, необходимые для реализации сериализации.*

Ключевые слова*: сериализация, персистентность, поток, объект, класс, интерфейс.*

Сериализация – это способность объекта сохранять полную копию его и любых других объектов, на которые он ссылается, используя поток вывода (например, внешний файл). Таким образом, объект может быть воссоздан из сериализованной (сохраненной) копии немного позже, когда это потребуется.

Сериализация впервые была введена в Java Development Kit версии 1.1. Она предоставляет собой метод для преобразования различных объектов или групп объектов, в битовые потоки или байтовые массивы, для дальнейшего сохранения или сетевой отправки. Полученные таким методом битовые потоки или байтовые массивы, возможно вернуть обратно в Java объекты. Непосредственно процесс сериализации проходит автоматически в классах ObjectInputStream и ObjectOutputStream. Реализовать данную функцию можно в процессе создания класса используя интерфейса Serializable. Процесс сериализиции также известен как маршалинг объекта, десериализация же известна как демаршалинг. Сериализация – это механизм, позволяющий объекту сохранять свою копию и ссылки на

объекты, в внешний файл посредством класса ObjectOutputStream. Возможно сохранить различные структуры данных, диаграммы, объекты класса JFrame или любые другие объекты, независимо от их типа. Также, сериализация позволяет сохранить информацию о типе объекта, чтобы, при десериализации, точно воссоздать тип объекта, перед сериализацией. Итак, сериализация предоставляет следующие возможности:

– хранение объектов, или сохранение свойств объекта в внешний файл, на диск или в базу данных;

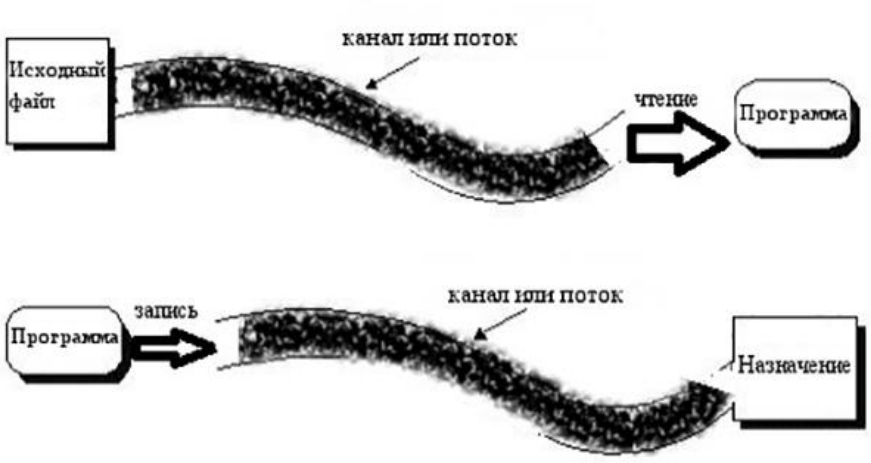
– вызовов удаленных процедур;

– распределение объектов, для примера, в программных компонентах типа COM, COBRA;

– идентификация изменений в данных переменных во времени;

– для полного понимания концепции сериализации, необходимо четко понимать две другие концепции, такие как персистентность потоков и объектов;

– у любой программы должна быть способность записывать информацию в внешний файл или поток, а также быть способной считывать эту информацию из места ее хранения. В Java, каналы для записи и считывания данных, называются потоками или stream (рис. 1).



-

Рис. 1. Представление Потоков

Потоки в основном принадлежат двум типам классов:

– Streams;

– Reader и Writer.

В каждом потоке для записи данных, содержится набор методов записи, а также набор для чтения. При создании потока все подобные наборы методов должны быть определены и вызваны.

Персистентность объекта это способность объекта существовать в независимости от времени жизни приложения. Это означает, что после того как объект перестает использоваться, встроенный мусорщик уничтожает данный объект, но способность персистентности позволяет такому объекту переживать чистки мусорщика, что дает возможность в дальнейшем при запуске приложения обратится к ним. Как способ реализации персистентность – это сохранение объектов в базу данных или внешний файл, а затем восстановление объектов в их состояние до сохранения, при помощи использованных мест хранения. Для это процесса и необходима сериализация. С ее помощью объект преобразуется, как например в битовый поток, для последующего сохранения в внешний файл.

Для сериализации объектов класс должен реализовывать интерфейс java.io.Serializable. У интерфейса Serializable нет методов, все что он делает это помечает класс, для того чтобы можно было его идентифицировать, как сериализуемый. Только у подобного сериализованного класса поля объекта могут быть сохранены. Стоит учесть, что методы или конструкторы не сохраняются, как части сериализованного потока. Если какой-либо объект действует как ссылка на другой объект, то поля этого объекта также сериализованны, если класс этого объекта реализует интерфейс Serializable. Другими словам, получаемый таким образом граф этого объекта, сериализуем полностью. Граф объекта включает дерево или структуру полей объекта и его подобъектов.

Особенность сериализации объектов использована во многих распределенных системах, как способ передачи данных. Но сериализация раскрывает скрытые детали, таким образом разрушая подлинность абстрактных типов данных, что в свою очередь разрушает инкапсуляцию. В то же время приятно знать, что данные сериализованного объекта, те же самые данные, что были в исходном,

оригинальном объекте. Это также отличная возможность для реализации интерфейса ObjectInputValidation и переопределения метода validateObject(), даже если используются несколько строк кода.

Список литературы

1. Брюс Э. Философия Java. – Питер, 2009. – С. 289–291.

2. Документация Java Standard Edition [Электронный ресурс]. – Режим доступа: https://docs.oracle.com/javase/8/docs/

3. Нимейер П. Программирование на Java. Исчерпывающее руководство для профессионалов / П. Нимейер, Д. Леук. – 2014. – С. 609–611.

4. Сериализация и десериализация в Java [Электронный ресурс]. – Режим доступа: http://www.ccfit.nsu.ru/~deviv/courses/oop/java_ser_rus.html