**Сериализация. Начало.**

[Java Core](https://javarush.ru/quests/QUEST_JAVA_CORE)

[Уровень 10](https://javarush.ru/quests/lectures?quest=QUEST_JAVA_CORE&level=10), Лекция 3

— Помнишь, мы сегодня разбирали сохранение объектов в файл и чтение из файла?

— Да, только мы сохраняли в поток вывода, а читали из потока ввода.

— Молодец, Амиго. Приятно слышать, что ты замечаешь такие мелочи. А ты бы смог дописать код, чтобы было сохранение в файл и чтение из файла?

— А что там писать?! Объявил FileInputStream и FileOutputStream и передавай их в методы save & load. Тут уже ничего не перепутаешь — все просто.

— Рада за тебя. Итак, новая тема – **сериализация**.

**Сериализация** – это практически то же самое, что мы с тобой только что делали, только гораздо круче и встроено прямо в Java-машину. Java-машина умеет сохранять и загружать свои объекты. Для этого ей даже не требуются методы save & load: все объекты хранятся внутри Java-машины, и она имеет к ним полный доступ.

Мы просто берем объект и сохраняем его в поток/читаем из потока:

Код

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

{

Cat cat = new Cat();

//save cat to file

FileOutputStream fileOutput = new FileOutputStream("cat.dat");

ObjectOutputStream outputStream = new ObjectOutputStream(fileOutput);

outputStream.writeObject(cat);

fileOutput.close();

outputStream.close();

//load cat from file

FileInputStream fiStream = new FileInputStream("cat.dat");

ObjectInputStream objectStream = new ObjectInputStream(fiStream);

Object object = objectStream.readObject();

fiStream.close();

objectStream.close();

Cat newCat = (Cat)object;

}

— И все?

— Да. Там очень большой и сложный механизм сериализации, который поддерживает сохранение в поток и чтение из потока почти всех типов данных.

— Почти всех, это значит не всех?

— Да, дело в том, что не все объекты по своей сути можно сохранить. Некоторые объекты не хранят все свои данные в себе, а лишь ссылаются на другие объекты и/или источники данных. Например, консоль (System.in), поток ввода (InputStream), или что-нибудь еще.

Поэтому разработчики Java придумали специальный интерфейс-маркер – **Serializable**. Его называют **маркером**, т.к. он не содержит никаких данных и методов. Он используется только для того, чтобы «помечать» (маркировать) классы. Если мы считаем, что наш класс хранит в себе все свои данные, тогда мы можем пометить его этим маркером – написать **implements Serializable**.

Пример «кота» с поддержкой сериализации:

Код

class Cat implements Serializable

{

public String name;

public int age;

public int weight;

}

Когда мы пытаемся сериализовать (сохранить) какой-нибудь объект, Java-машина проверяет – поддерживает ли он сериализацию: реализует ли он интерфейс Serializable? Если да, то сохраняет объект, если нет – выкидывает исключение о невозможности сериализации.  
Тут нужно понимать, что сериализуемый объект должен состоять тоже только из сериализуемых объектов.

— Ну, этого и следовало ожидать. Нельзя же сохранить целое, не сохранив какие-то его части.

— Именно так.

— А как же типы int, String, ArrayList?

— Они все поддерживают сериализацию, на этот счет разработчики Java специально позаботились. Тут проблем быть не должно.

Более того, при сериализации объекта сохраняется его тип. Теперь ты можешь в переменную класса с типом Object сохранить ссылку на объект Cat, и все это отлично сериализуется и десериализуется.

— Десериализуется?

— Десериализация – так называют процесс, обратный сериализации – чтение и восстановление объекта из потока/файла.

— Тогда вопросов больше нет.