**Сохранение массива объектов**

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

[Уровень 10](https://javarush.ru/quests/lectures?quest=QUEST_JAVA_CORE&level=10), Лекция 1

— Привет, Амиго! Сегодня мы познакомимся с еще одной интересной темой. А именно: сохранением и загрузкой (восстановлением) объектов. Допустим у нас есть класс Cat:

Код

class Cat

{

public String name;

public int age;

public int weight;

}

И мы хотим добавить в него удобный механизм сохранения в файл и загрузки из файла.

Вот как это можно сделать:

Код

class Cat {

public String name;

public int age;

public int weight;

public void save(PrintWriter writer) throws Exception {

writer.println(name);

writer.println(age);

writer.println(weight);

writer.flush();

}

public void load(BufferedReader reader) throws Exception {

name = reader.readLine();

age = Integer.parseInt(reader.readLine());

weight = Integer.parseInt(reader.readLine());

}

}

— О! Это же очень просто. Мы просто печатаем значения всех аргументов, по одному в каждой строчке. А при загрузке читаем их в том же порядке. Отличное решение.

— Спасибо, Амиго. А можешь написать, как будут выглядеть методы save и load у такой группы классов:

Код

class Cat

{

public String name;

public int age;

public int weight;

}

class Dog

{

public String name;

public int age;

}

class Human

{

public Cat cat;

public Dog dog;

}

У тебя есть объект человек, и он может иметь одну собаку и одного кота.

— У меня есть решение:

Код

class Cat {

public String name;

public int age;

public int weight;

public void save(PrintWriter writer) throws Exception {

writer.println(name);

writer.println(age);

writer.println(weight);

writer.flush();

}

public void load(BufferedReader reader) throws Exception {

name = reader.readLine();

age = Integer.parseInt(reader.readLine());

weight = Integer.parseInt(reader.readLine());

}

}

Код

class Dog {

public String name;

public int age;

public void save(PrintWriter writer) throws Exception {

writer.println(name);

writer.println(age);

writer.flush();

}

public void load(BufferedReader reader) throws Exception {

name = reader.readLine();

age = Integer.parseInt(reader.readLine());

}

}

Код

public class Human {

public Cat cat;

public Dog dog;

public void save(PrintWriter writer) throws Exception {

cat.save(writer);

dog.save(writer);

}

public void load(BufferedReader reader) throws Exception {

cat.load(reader);

dog.load(reader);

}

}

— Очень хорошее решение. Но что будет, если у человека нет кота, а есть только собака?

Где проверки на null?

— Сейчас исправлю:

Код

public class Human {

public Cat cat;

public Dog dog;

public void save(PrintWriter writer) throws Exception {

if (cat != null)

cat.save(writer);

if (dog != null)

dog.save(writer);

}

public void load(BufferedReader reader) throws Exception {

cat = new Cat();

cat.load(reader);

dog = new Dog();

dog.load(reader);

}

}

— Все равно не очень верно. У тебя две ошибки:

**1)** Если у человека нет ни кота, ни собаки, они все равно будут созданы, при вызове метода load

**2)** Если мы сохраним только собаку, то ее данные будут прочитаны котом при загрузке.

— А что же делать?

— Мы не можем пропускать запись переменных, иначе это вызовет сбой при чтении. Но нужно сделать так, чтобы переменные, чье значение было null при сохранении и после загрузки, получали null. Вот мой вариант:

Код

public class Human {

public Cat cat;

public Dog dog;

public void save(PrintWriter writer) throws Exception {

String isCatPresent = cat != null ? "yes" : "no";

writer.println(isCatPresent);

writer.flush();

if (cat != null)

cat.save(writer);

String isDogPresent = dog != null ? "yes" : "no";

writer.println(isDogPresent);

writer.flush();

if (dog != null)

dog.save(writer);

}

public void load(BufferedReader reader) throws Exception {

String isCatPresent = reader.readLine();

if (isCatPresent.equals("yes")) {

cat = new Cat();

cat.load(reader);

}

String isDogPresent = reader.readLine();

if (isDogPresent.equals("yes")) {

dog = new Dog();

dog.load(reader);

}

}

}

— Да, мне нравится такое решение.

— Да, что-то в нем есть.