14. - Soubory a serializace – Ukládání a načítání dat, formáty souborů

Co je to soubor?

Soubor jako takový je jenom pojmenovaná skupinka nějakých dat. Data můžou být různého typu, obrazové, textové, zvukové, programové… Soubory se používají k organizaci dat a mohou být jakkoliv upravovány, čteny a posílány dál. Každý soubor má nějaké jméno, které následně slouží k identifikace a adresování, kde jsou v paměti data uložená.

Java pro práci se soubory využívá knihovnu java.IO – Input Output.

Čtení ze souboru je thread-safe, ale zápis thread-safe není a musí se používat například RWL – Read Write Lock Conccurent patterny.

Co je to serializace?

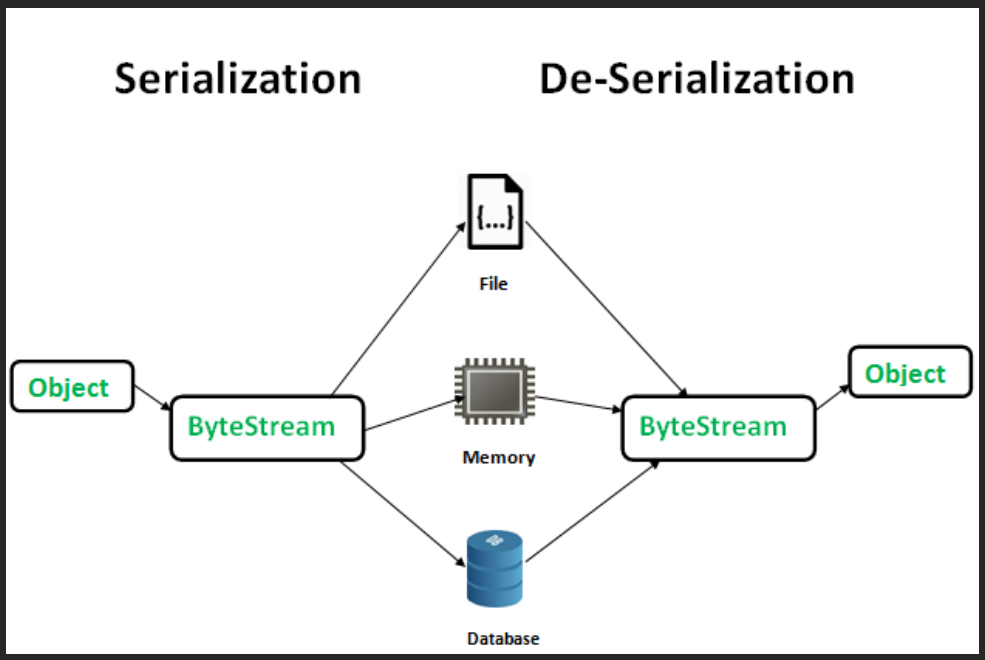
Serializace je nějaký proces, převod objektů a nebo datových struktur do formátu, který bude jednoduše přenositelný. Typicky se využívá například pro ukládání stavu aplikace, ukládání a načtení konfiguračních souborů a nebo obyčejný přenos dat mezi procesy v počítači. Během serializace jsou data přenášena jako série bytů, které se následně zase deserializují do původní podoby.

Data, která jsou totiž po vypnutí programu na stacku nebo na heapu, se ztratí. Serializace nám umožňuje tyto objekty nebo datové struktury zachovat.

Java jako taková pro serializaci používá mechanismus Java Serialization, který umožňuje ukládat objekty ja disk nebo je přenášet přes síť.

Je spoustu různých formátů pro serializaci, nejznámější z nich jsou například XML a nebo JSON.

Ukládání a načítání dat



Serializování objektů

Ukládání dat funguje poměrně jednoduše. V Javě, pokud chceme nějaký objekt uložit, implementujeme do třídy java.io.Serializable. Pozor, je vždy serializován celý objekt. Můžeme to obejít tím, že proměnné dáme klíčové slovo „`transient`“. Tím říkáme, že se proměnná nemá serializovat. Pokud nás zajímá, jestli je něco serializovatelné nebo ne, podíváme se jenom do dokumentace. Pokud implementuje knihovni java.io.Serializable, tak ano, jinak ne.

K serializování objektů se využívá ObjectOutputStream – Stream je proud dat v bytové podobě. Pomocí metody .writeObject se následně objekt serializuje.

Důležité je mít také FileOutputStream – ten totiž vytvoří Stream a ObjectOutputStream tento výstupní Stream zabalí tak, aby se vědělo, že se jedná o serializovaný objekt.

V Javě se standartně serializovaným objektům dává koncovka .ser

```
Employee e = new Employee();
e.name = "Reyan Ali";
e.address = "Phokka Kuan, Ambehta Peer";
e.SSN = 11122333;
e.number = 101;

try {
   FileOutputStream fileOut = new FileOutputStream("src\\employee.ser");
   ObjectOutputStream out = new ObjectOutputStream(fileOut);
   out.writeObject(e);
   out.close();
   fileOut.close();
   System.out.printf("Data jsou serializována v src\\\\employee.ser");
} catch (IOException i) {
   i.printStackTrace();
}
```

V tomto případě do souboru src/employee.ser uložíme binární reprezentaci objektu e, tedy zaměstnance. Při deserializaci vytvoříme novou instanci Employee a ta bude obsahovat všechny hodnoty, které měl náš Employee před deserializací.

Je důležité poznamenat, že podobně jako při serializaci, při deserializaci se musí zachovat stejný datový typ a pořadí prvků, jinak by mohlo dojít k chybě.

Deserializace

V případě deserializace my načítáme postupně binární informace o objektu pomocí FileInputStream a ObjectInputStream. V tomto případě my využijeme metodu readObject, který má ObjectInputStream a přetypujeme informace z této metody do našeho Zaměstnance.

```
Employee e = null;
try {
    FileInputStream fileIn = new FileInputStream("/tmp/employee.ser");
    ObjectInputStream in = new ObjectInputStream(fileIn);
    e = (Employee) in.readObject();
    in.close();
    fileIn.close();
} catch (IOException i) {
    i.printStackTrace();
    return;
} catch (ClassNotFoundException c) {
    System.out.println("Employee class not found");
    c.printStackTrace();
    return;
}
```

Důležité je samozřejmě mít vše ošetřené v try – catch blocku, který kontroluje případné problémy se čtením souboru, jako například ty, že soubor neexistuje, nebo to, že není možné vytvořit zaměstnance, protože třída neexistuje.

Pokud také je nějaké proměnná, kterou jsme neserializovali kvůli klíčovému slovu „`transient`“, nastaví se na defaultní hodnotu. V případě nějaékho čísla se bude jednat o nulu, v případě booleanu zase false.

Formáty souborů

Formát souboru nám říká, co můžeme v souboru najít. Je jich poměrně hodně a každý je rozdílný. Všechny soubory jsou ve skutečnosti binárními soubory, protože počítač je ukládá jako posloupnost bitů.

Textové soubory

Textové soubory jsou takové soubory, které v sobě mají text, kterému můžeme člověk při otevření rozumnět. Pokud bychom otevřeli binární soubor, nebudeme vědět, na co se koukáme. Příklad textového souboru může být právě třeba Word, textový soubor, nějaký kód nebo konfigurační soubory. Jde o to, že pokud obsah souboru přečteme, dostaneme třeba zpátky třeba „Kočičkyy“

Binární soubory

Binární soubory jsou takové soubory, které uchovávají binární informace. Nejsou čitelné lidským okem a jsou skládány do nějaékho formátu pro účely kompilace nebo interpretace. Vyznačují se také tím, že po přečtení souboru dostane přesně ty samá data, jako se nachází v souboru a nejsou žádné konverze, jako u textových, kde se následně bajty dekódují.

V případě binárních souborů, pokud bychom je přečetli, dostaneme změť náhodných znaků, nebo náhodných bajtů dat, třeba

Příkladem binárního souboru může být právě serializovaný soubor.

Zvukové a video formáty

Zvukové formáty jsou speciální druh binárních souborů. Data můžou být v komprimované i nekomprimovaném formátu. Tyto soubory obsahují informace o tom, jaké hodnoty amplitudy se mění v čase. Jsou uloženy jako binární čísla.