18. Soubory a serializace

Soubory

```
Data, které jsou používána při běhu programu na stacku a heapu jsou v OP a
po vypnutí programu se ztratí.
Pokud se chcem, aby data byla trvalá (perzintentní) a bylo je možné při
opětovném zapnutí programu znova načíst, tak se musí ukládat mimo zdrojový
kód, třeba do souborů nebo do databáze.
K souborům je přístup delší než k datům v OP.
Soubory se rozdělují na:
- Textové soubory s plochou strukturou (txt, csv)
- Textové soubory s vnitřní hierarchií (xml, json)
- Soubory binárního typu
- Databáze

Windows zakazuje zápis na systémový oddíl disku, musí být povolen
administrátorem nebo se nejedná o zápis do složek uživatele.
Také antivirový program může vyhodit jako škodlivý a také ho zablokuje.
Stream = datový proud, přes tento proud lze číst a zapisovat data.

IOException obsahuje výjimka na práci se soubory, které by bylo dobré
ohlídat: DirectoryNotFoundException, DriveNotFoundException,
FileLoadException, FileNotFoundException.
```

Třídy k práci se soubory

```
File
Třída určená k práci se soubory, složkami nebo jinými typy souborů.
Umožní nám zjišťovat jestli
fileExists,find,deleteIfExists,getOwner,getLastModifiedTime,isWritable,isSa
meFile,isRegularFile,isReadable,isHidden,atd.

FileWriter
Zapíše text do znakových souborů za použití defaultní šířky bufferu.
Zakódování znaků na byty používá buď speciální kódovací sadu nebo
platformovou defaultní sadu.
File Writer se používá na zapisování znaků . Na zapisování bytů je lépe
použít FileOutputStream.
```

```
FileReader
Třída na čtení znakových souborů.Pokud bychom chtěli nastavit kódování
znaků nebo šířku bufferu musíte vytvořit InputStreamReader nebo
FileInputStream.
```

Path
Třída na nalezení souboru v souborovém systému.Reprezentuje systémovou cestu.
Path reprezentuje hierarchickou sekvenci složek a názvů souborů oddělených speciálním separátorem nebo delimiterem.

Čtení ze souborů je možné thread-safe, tím pádem může číst soubor více vláken, ale zápis jednoduše nelze udělat (můžou se použít lockery, nebo thread-safe kolekce, která by sloužila jako dočasná úschovna dat, které by měla být zapsána a jedno vlákno by z ní zapisovalo)

Serializace

Pomáhá při uchování objektů do souborů, databáze,…
Převede objekt z OP na streamy bytů a ty jsou uloženy do nějakého souboru.
Jednoduchý kód na předvedení serializace a deserializaci

```
class Car implements Serializable {
 private static final long serialVersionUID = 1L;
 public String plate;

 public Car(String plate) {
  this.plate = plate;
 }
}
public class SerializationTest {
 private static void Serialize(Car a) {
  try {
   FileOutputStream fileOut = new FileOutputStream("car.ser");
   ObjectOutputStream out = new ObjectOutputStream(fileOut);
   out.writeObject(a);
   out.close();
   fileOut.close();
  } catch (IOException i) {
   i.printStackTrace();
  }
 }
 private static Car Deserialize() {
  Car e = null;
  try {
          FileInputStream fileIn = new FileInputStream("car.ser");
          ObjectInputStream in = new ObjectInputStream(fileIn);
          e = (Car) in.readObject();
          in.close();
          fileIn.close();
       } catch (IOException i) {
          i.printStackTrace();
       } catch (ClassNotFoundException c) {
          System.out.println("Employee class not found");
```

```
            c.printStackTrace();
        }
   return e;
  }
  public static void main(String[] args) {
   Car a = new Car("adfasdfasf");
   Serialize(a);
   Car b = Deserialize();
   System.out.println(b.plate);
  }
}
```

Deserializace

Je to opak serializace, převede data ze souborů na objekty v OP.

Je to užitečné při přenosů dat jako soubory mezi počítači, třeba po síti. Nebo při ukládání nastavení aplikace, které by bylo vytvořeno objektově a chtělo by se uchovávat do dalšího spuštění aplikace.
Jsou možnosti jak serializovat objekty, do binárního typu, ty nelze jednoduše číst anebo do CSV, XML nebo JSON
Objekt, který chcem serializovat musí mít konstruktor bez vstupních parametrů (můžou být 2 jeden normální s parametry a druhý prázdný pro serializaci), protože deserializer nejdříve vytvoří novou instanci a poté jí nastavuje vlastnosti jak je čte ze souboru.