18. Práce se soubory

Účel třídy `File`

- Třída **File** v Javě slouží ke zpracování souborů a adresářů v souborovém systému
- Účelem této třídy je poskytnout programátorům možnost pracovat se soubory a adresáři na úrovni operačního systému
- Poskytuje řadu metod pro manipulaci se soubory a adresáři
    - **exists()** - kontroluje, zda soubor nebo adresář existuje
    - **createNewFile()** - vytvoří nový soubor
    - **delete()** - odstraní soubor nebo adresář
    - **listFiles()** - vrátí seznam souborů a adresářů v daném adresáři
    - **isDirectory()** - kontroluje, zda je daný soubor adresářem

Kódování znaků a možné problémy při čtení (parsování)

- Kódování znaků se týká reprezentace znaků v počítači. Znak je reprezentován binárním kódem, který odpovídá číselné hodnotě. Existují různé kódování znaků, jako například ASCII, UTF-8
- Při čtení (parsování) souborů může dojít k problémům, pokud se **kódování souboru neshoduje s kódováním, které používáme pro jeho čtení**. Pokud se to stane, může se stát, že některé znaky budou špatně interpretovány nebo zcela ztraceny
- Problémy s kódováním mohou také vzniknout, pokud se **při čtení použije nesprávný počet bajtů na jeden znak**

Význam bufferování

- Ukládání dat do paměti
- V kontextu čtení a zápisu souborů znamená bufferování, že se data zapisují do paměti (bufferu) místo přímého zápisu na disk. Tím se snižuje počet operací s diskem, což může zlepšit výkon aplikace
- Například, pokud máme soubor o velikosti 10 MB a čteme ho po jednom bytu, každé čtení vyžaduje přístup k disku. Pokud použijeme bufferování, čtení se provádí většími bloky, které jsou uloženy v paměti, což znamená méně přístupů na disk a zvýšenou efektivitu
- V Javě se pro bufferování čtení a zápisu souborů používají třídy **BufferedReader** a **BufferedWriter**. Tyto třídy ukládají data do paměti, dokud se nenaplní buffer, a pak provádějí operace s diskem

Appending – význam při psaní do souboru

- **Nová data jsou přidána na konec již existujícího souboru**, místo toho, aby se původní soubor přepsal novými daty. To znamená, že původní data zůstanou v souboru nedotčena a nová data se přidají za ně
- V Javě můžeme použít režim appending při zápisu do souboru pomocí třídy **FileWriter**
- K tomu použijeme konstruktor, který má **druhý parametr s hodnotou true**

```java
File file = new File("example.txt");
try (FileWriter writer = new FileWriter(file, true)) {
    writer.write("nová data");
} catch (IOException e) {
    // zpracování výjimky
}
```

Významné operace při čtení a psaní do souborů

- Otevření souboru
  - K otevření souboru v Javě používáme třídy jako File, FileReader, FileWriter, BufferedReader, BufferedWriter, apod. Vytváření instancí těchto tříd může být prováděno s parametry, jako je jméno souboru nebo cesta k souboru
- Čtení souboru
  - K čtení souboru v Javě používáme třídy jako FileReader, BufferedReader, Scanner, apod. Tyto třídy umožňují čtení souboru po řádcích nebo po znacích, nebo dokonce čtení souboru jako celku
- Zápis do souboru
  - K zápisu do souboru v Javě používáme třídy jako FileWriter, BufferedWriter, PrintWriter, apod. Tyto třídy umožňují zápis textových dat do souboru, přičemž můžeme specifikovat, zda chceme přepsat soubor novými daty nebo přidat nová data na konec již existujícího souboru
- Uzavření souboru
  - Po dokončení čtení nebo zápisu do souboru musíme soubor uzavřít, aby byl uvolněn pro další použití. To lze provést voláním metody **close()** na instanci třídy, kterou jsme použili k otevření souboru
- Správa výjimek
  - Při práci se soubory v Javě musíme být připraveni na výjimky, které mohou nastat během čtení nebo zápisu. Tyto výjimky mohou být způsobeny např. špatným formátem souboru, problémy s diskem nebo nedostatečnými oprávněními pro přístup k souboru. Proto je důležité správně zacházet s výjimkami pomocí try-catch

Prakticky:

- Představení možnosti čtení textu ze souboru, velikost souboru, popis kódu

```java
File file = new File("nazev_souboru.txt");
long velikost = file.length();
System.out.println("Velikost souboru je: " + velikost + " bajtů");
```