25 - Zpracování a parsování textových dat, regulární výrazy, kódování a stringy.

Co jsou to textová data?

Textová data jsou informace zachované jako text – string. String jako takový je jen řetězec charakterů – znaků. String je v Javě objekt, takže má různé funkce, jako je například `.size()`, `.chatAt()`, `.substring()`, `.replace()` nebo `.toUpperCase()`

String má tolik byte, kolik má počet znaků krát dva. Je to způsobené tím, že Java využívá Unicode kódování, tudíž je možné zakódovat 65536 různých znaků. Paměť spotřebovaná na obsah stringu s 10 znaky proto bude 20 bytů.

Stringy jsou v Javě ukládány v Method Aree na String pool (Pokud nejsou vytvořeny pomocí „new“, poté jdou na Heap)

Stringy jako takové nejde měnit, místo toho vytváříme nový objekt s novým obsahem.
Existuje String Builder, který umožňuje měnit vysloveně String a manipulovat s místem v paměti tak, aby se nemusela vytvářet nová instance třídy. String Buffer je vláknově bezpečný, lze používat v různých vláken bez nutnosti řešit synchronizaci. V novějších verzích Javy se ještě využívá StringBuilder, který není vláknově bezpečný, ale někdy místo toho může být rychlejší.

Zpracování a parsování textových dat

Zpracování textových řetězců

Jedná se o manipulace s daty, díky kterým například vyhledáváme řetězce v řetězcích, nebo nahrazujeme určité znaky pomocí oddělovače, spojujeme nebo je dělíme a nebo je formátujeme. Můžeme na to použít mnoho metod.

Je spousta metod, díky kterým budeme moci pracovat a zpracovávat textové řetězce. Například:

Metoda `replace()` - Tato metoda nahradí všechny výskyty určitého řetězce jiným řetězcem.

Metoda `replaceAll()` - Tato metoda nahradí všechny výskyty určitého regulárního výrazu jiným řetězcem.

Metoda `split()` - Tato metoda rozdělí řetězec na podřetězce na základě určitého oddělovače.

Metoda `trim()` - Tato metoda odstraní mezery z počátku a konce řetězce.

Metoda `indexOf()` - Tato metoda vrátí index prvního výskytu určitého slova / řetězci v jiném řetězci.

Metoda `charAt()` - Tato metoda vrátí znak na zadané pozici v řetězci.

Metoda `substring()` - Tato metoda vrátí podřetězec z řetězce na základě zadaných indexů.

Parsování textových řetězců

Znamená vysloveně extrakce určitých dat z těchto řetězců. Cílem parsování je získat nějaké informace z textu, třeba jméno, číslo, datum, podobné věci. Můžeme k tomu třeba použít i známý regex.

V Javě je možné parsovat texty pomocí metod `parseInt(), parseFloat(), parseDouble(), parseBoolean()` a podobné, které ze stringu udělají jiný datový typ, v případě, že se nedaří, vyhodí se vyjímka.

Regulární výrazy

Regulární výrazy nám umožňují zkontrolovat, jestli daný řetězec obsahuje nebo neobsahuje určitý pattern. Využíváme je převážně k tomu, abychom ošetřili uživatelův vstup – třeba, jestli zadal jméno v rozsahu tří až šestnácti znaků, jestli má heslo jeden speciální znak, jestli má telefonní číslo 9 čísel a správnou předvolbu…

Regulární výrazy se používají v mnoha programovacích jazycích, včetně Javy. Regulárné výrazy mají určité charakteristiky, jako například zápis, poté posloupnost oněch znaků, speciální operátory, které definují vztahy mezi skupinami, které jsou zase ve hranatých závorkách…

V Javě importujeme pomocí
`java.util.regex.Pattern` a `java.util.regex.Matcher`

Zde je příklad regexu, který kontroluje nejdříve jméno, devatenáct znaků dlouhé, poté jedna pomlčka a poté čtyři čísla

```
String regex = "^[A-Za-z]{1,19}-\\d{4}$";
```

^ - Začátek řetězce

[A-Za-z] - Skupina obsahující jenom velká a malá písmena, a-z a A-Z

{1,19} - Kvantifikátor, který specifikuje, že znaková skupina musí být přítomna nejméně 1 a nejvýše 19krát.

- - Pomlčka

\\d – Specifikuje, že se jedná o skupinu čísel

{4} – Kvantifikátor, říká, že musí být přesně čtyři čísla.

$ - Konec řetězce

Je nekonečno různých příkladů a spoustu možností, jak regulární výrazy využít a zapsat

Pattern

Pattern je právě regulérní výraz, kterým budeme ověřovat, zdali nějaký řetězec splňuje požadovaný tvar pomocí Matcheru

Matcher

Matcher nám porovnává regulérní výraz a nějaký textový řetězec. Jedná se o objekt, který má spousty metod na manipulaci s regulérním výrazem. Matcher se dává do while, který projíždí celý string a v případě, že něco najde, nastane nějaká část kódu. Můžeme třeba udělat:

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class MatcherExample {
    public static void main(String[] args) {
        String inputString = "The quick brown fox jumps over the lazy dog";
        String regex = "fox|dog";

        Pattern pattern = Pattern.compile(regex);
        Matcher matcher = pattern.matcher(inputString);

        while(matcher.find()) {
            System.out.println("Found match: " + matcher.group());
            System.out.println("Starting index: " + matcher.start());
            System.out.println("Ending index: " + matcher.end());
        }
    }
}
```

V tomto případě by nastal matcher.find() dvakrát – poprvé u fox, podruhé u dog.

Kódování

Kódování je způsob, jakým jsou nějaké znaky a symboly převedeny na formu, kterou bude počítač schopný zpracovat. Většinou se k tomu používají přesně stanovené tabulky a znakové sady.

Neznámější z nich jsou například:

ASCII

ASCII je **sedmibitová** tabulka, kde je každému písmenu v anglické abecedě a různým dalším znakům přiřazeno nějaké číslo. Bylo vytvořeno převážně pro anglicky mluvící země, proto nebyla potřeba mít bytové. ASCII využívá 0-127 bitů.

UNICODE

Unicode přiřazuje každému znaku unikátní kód. Jeho kódování podporuje všechny možné jazyky, texty, písma včetně cyrilice, arabštiny, čínštiny a mnohem další. Má také dvou bytové kódování, čímž má možnost využít dohromady 65536 znaků.

UTF-8

UTF-8 umožňuje reprezentovat všechny UNICODE znaky pomocí jednobytových a vícebytových sekvencí. Znaky pod 128 jsou kódovány jako ASCII, znaky mezi 128-2047 jsou kódovány dvojicí bajtů, 2048-65535 se kódují pomocí trojice bajtů a cokoli vnad 65536 se kóduje pomocí více bajtů. Je tedy proměnlivý počet bajtů.

Aby se dalo ale poznat, jaký znak má kolik bajtů (Protože si nemůžete být jistí, jestli čtyři byty jsou čtyř písmena, dvě nebo jedno písmena), rozeznává se podle následujících pravidel.

Začíná-li byte 110 – Jedná se o dvoubytový znak.
Bude-li byte 1110 – Jedná se o tříbytový znak
Bude-li byte 11110 – Jedná se o čtyřbytový znak.

UTF-16

Funguje stejně jako UTF-8, ale využívá místo toho striktně 16 bitů. Tudíž i ASCII znaky jsou zakódovány také na 2 byty, UNICODE stejně.

Microsoft - CP1250

CP1250 obsahuje 256 znaků, tudíž jeden byte, a podporuje češtinu, polštinu a další středoevropské jazyky. ASCII se zapisuje jako ASCII, nicméně CP1250 přidává i háčky a čárky pro diakritiku.