**Associative arrays (Maps)**

**Maps - cheat sheet**

*1. Видове*

а) **HashMap** -> редът на записите не е гарантиран

b) **LinkedHashMap** -> редът на поставянето на записите се запазва (първия поставен запис си остава на първо място)

c) **TreeMap** -> записите се сортират по ключ в лексикографски ред (ако ключовете са числа – в ascending order, ако ключовете са тесктове/символи – от a към z)

*2. Основни функционалности*

a) **containsKey(key)** -> проверява дали в map-а има запис с такъв ключ -> резултат true ако има, false ако няма

b) **containsValue(value)** -> проверява дали в map-а има запис с такова value -> резултат true ако има, false ако няма

c) **keySet()** -> връща всички ключове от всички записи

d) **entrySet()** -> връща всички записи

e) **values()** -> връща всички value-та от всички записи

f) **get(key)** -> връща стойността, която стои срещу дадения ключ

g) **size()** -> връща броя на записите

h**) put(key, value)** -> добавя запис с дадения ключ и стойност

i) **putIfAbsent(key, value)** -> добавя запис с дадения ключ и стойност, ако вече няма запис с такъв ключ

j) **clear()** -> премахва всички записи от map-a

k) **remove(key)** -> премахва запис с дадения ключ

l) **remove(key, value)** -> премахва запис с дадения ключ и стойност

**Iterating Through Map**

* Iterate through objects of type Map.Entry<K, V>
* Cannot modify the collection (read-only)

*Map<String, Double> fruits = new LinkedHashMap<>();*

*fruits.put("banana", 2.20);*

*fruits.put("kiwi", 4.50);*

*for (Map.Entry<K, V> entry : fruits.entrySet()) {*

*System.out.printf("%s -> %.2f%n",*

*entry.getKey(), entry.getValue());* ***// entry.getKey() -> fruit name***

***entry.getValue() -> fruit price***

*}*

**Text processing**

**String**

* **Accessible by index (read-only)**

*String str = "Hello, Java";*

*char ch = str.charAt(2); // l*

**Initializing a String**

* **Converting a string from and to a char array:**

*String str = new String(new char[] {'s', 't', 'r'});*

*char[] charArr = str.toCharArray();*

*// ['s', 't', 'r']*

**Joining Strings**

* **String.join("", …) concatenates strings** *String t = String.join("", "con", "ca", "ten", "ate"); // "concatenate"*

**Substring**

* **substring(int startIndex, int endIndex)** *String card = "10C";*

*String power = card.substring(0, 2);*

*System.out.println(power); // 10*

* **substring(int startIndex)** *String text = "My name is John";*

*String extractWord = text.substring(11);*

*System.out.println(extractWord); // John*

**Searching**

* **indexOf() - returns the first match index or -1** *String fruits = "banana, apple, kiwi, banana, apple";*

*System.out.println(fruits.indexOf("banana")); // 0*

*System.out.println(fruits.indexOf("orange")); // -1*

* **lastIndexOf() - finds the last occurrence** *String fruits = "banana, apple, kiwi, banana, apple";*

*System.out.println(fruits.lastIndexOf("banana")); // 21*

*System.out.println(fruits.lastIndexOf("orange")); // -1*

* **contains() - checks whether one string contains another**

*String text = "I love fruits.";*

*System.out.println(text.contains("fruits")); // true*

*System.out.println(text.contains("banana")); // false*

**Splitting**

* **Split a string by a given pattern**

*String text = "Hello, john@softuni.bg, you have been*

*using john@softuni.bg in your registration";*

*String[] words = text.split(", ");*

*// words[]: "Hello", "john@softuni.bg","you have been…"*

* **Split by multiple separators** *String text = "Hello, I am John.";*

*String[] words = text.split("[, .]+"); // "Hello", "I", "am", "John"*

**Replacing**

* **replace(match, replacement) – replaces all occurrences. The result is a new string (strings are immutable)**

*String text = "Hello, john@softuni.bg, you have been*

*using john@softuni.bg in your registration.";*

*String replacedText = text*

*.replace("john@softuni.bg", "john@softuni.com");*

*System.out.println(replacedText);*

*// Hello, john@softuni.com, you have been using*

*john@softuni.com in your registration.*

**StringBuilder**

* **Use the StringBuilder to build/modify strings** *StringBuilder sb = new StringBuilder();*

*sb.append("Hello, ");*

*sb.append("John! ");*

*sb.append("I sent you an email.");*

*System.out.println(sb.toString()); // Hello, John! I sent you an email.*

**StringBuilder Methods**

* **append() - appends the string representation of the argument**

*StringBuilder sb = new StringBuilder();*

*sb.append("Hello Peter, how are you?");*

* **length() - holds the length of the string in the buffer** sb.append("Hello Peter, how are you?");

System.out.println(sb.length()); // 25

* **setLength(0) - removes all characters**
* **charAt(int index) - returns char on index***StringBuilder sb = new StringBuilder();*

*sb.append("Hello Peter, how are you?");*

*System.out.println(sb.charAt(1)); // e*

* **insert(int index, String str) –** sb.insert(11, " Ivanov");

System.out.println(sb);

// Hello Peter Ivanov, how are you?

* **replace(int startIndex, int endIndex, String str) - replaces the chars in a substring**

*sb.append("Hello Peter, how are you?");*

*sb.replace(6, 11, "George");*

* **toString() - converts the value of this instance**

**to a String**

*String text = sb.toString();*

*System.out.println(text); // Hello George, how are you?*

**Regular Expressions**

Основен синтаксис:

[A-Z] - една главна буква (аски код от 65 до 90)

[a-z] - една малка буква (аски код от 97 до 120)

[0-9] - една цифра [0-9] (аски код от 48 до 57)

[aeiou] - всички гласни букви

[^aeiou] - всички съгласни букви

\w - един символ, който може да е малка буква, главна буква, цифра или долна черта

\W - един символ, различен от малка буква, главна буква, цифра или долна черта

\s - един интервал

\S - един символ, различен от интервал

\d - една цифра [0-9] (аски код от 48 до 57)

\D - един символ, различен от цифра

Брой на срещанията:

\* -> срещания 0 или безброй много пъти

+ -> срещания 1 или безброй много пъти

? -> срещания 0 или 1 пъти

{брой} -> срещания {брой} пъти

**Групиране:** (?<име на групата> regex)

(?<day>\d{2})-(?<month>\w{3})-(?<year>\d{4}) // 22-Jan-2015

**Използване в Java:**

String text = scanner.nextLine();

String regex = "\\b[A-Z][a-z]+ [A-Z][a-z]+\\b";

Pattern pattern = Pattern.compile(regex); // шаблон

Matcher matcher = pattern.matcher(text); //текстовете от променливата text, които отговарят на шаблона

* **find()** - gets the first pattern match

String text = "Andy: 123";

String pattern = "([A-Z][a-z]+): (?<number>\\d+)";

Pattern regex = Pattern.compile(pattern);

Matcher matcher = regex.matcher(text);

System.out.println(matcher.find()); // true

System.out.println(matcher.group()); // Andy: 123

System.out.println(matcher.group(0)); // Andy: 123

System.out.println(matcher.group(1)); // Andy

System.out.println(matcher.group(2)); // 123

System.out.println(matcher.group("number")); // 123

**Replacing with Regex**

* replaceAll(String replacement)
* replaceFirst(String replacement)

String regex = "[A-Za-z]+";

String string = "Hello Java";

Pattern pattern = Pattern.compile(regex);

Matcher matcher = pattern.matcher(string);

String res = matcher.replaceAll("hi"); // hi hi

String res2 = matcher.replaceFirst("hi"); // hi Java

**Splitting with Regex**

* split(String pattern) - splits the text by the pattern
* Returns String[]

String text = "1 2 3 4";

String pattern = "\\s+";

String[] tokens = text.split(pattern); // tokens = {"1", "2", "3", "4"}

**Solution: Match Dates** String input = reader.readLine();

String regex =

"\\b(?<day>\\d{2})(\\.|\\/|\\-)(?<month>[A-Z][az]{2})\\2(?<year>\\d{4})\\b";

Pattern pattern = Pattern.compile(regex);

Matcher matcher = pattern.matcher(dates);

while (matcher.find()) {

System.out.println(String.format("Day: %s, Month: %s, Year: %s",

matcher.group("day"), matcher.group("month"),

matcher.group("year")));

}

/(@|#)(?<firstWord>[A-Za-z]{3,})\1 – шаблон за: думата да започва с @ или # като буквите в думата да са 3 или повече, а 1 в края на шаблона показва, че с какъвто символ започва, с такъв трябва и да завършва