1. **BASIC SYNTAX, CONDITIONAL STATEMENTS AND LOOPS**
   1. **Четене от конзолата**

int age = Integer.parseInt(scanner.nextLine()); //стринг с преобразуване към инт

String username = scanner.nextLine(); // стринг

double pricePerCapsule = Double.parseDouble(scanner.nextLine()); // стринг към дабъл

* 1. **Преобразуване между типове данни**

Кастване : **char -> int**

char symbol = 'A';

int asciiValue1 = (int) symbol; // explicit casting, символ към аски код

int asciiValue2 = symbol; // implicit casting

String number = "145";

char symbol = number.charAt(0); //**String -> char**

String symbolAsText = symbol + ""; // **char -> String**

int digit = Integer.parseInt(symbolAsText); // **String -> int**

double digit = Double.parseDouble(symbolAsText); **String -> Double**

int digit = Integer.parseInt(inputNumber.charAt(position) + "") //**позиция от String -> int**

* 1. **Форматирано отпечатване**

System.out.printf("Name: %s, Age: %d, Grade: %.2f", name, age, averageGrade);

**\s – String**

**\d – int**

**\%.2f - double**

**\n – нов ред**

* 1. **Други**

group.equals("Students") // стрингът group е еднакъв на Students

!input.equals("Start") // стрингът input не е еднакъв със стрингът Start

number % 2 != 0 // числото не е четно, %2 – остатък при делене на две

int lastDigit = number % 10 // последна цифра на число

number /= 10; // премахване на последната цифра на число

if (Character.isUpperCase(symbol) // проверка дали стринг започва с главна буква

1. **МАСИВИ**
   1. **Обща информация**

//1. масивът е съвкупност от еднотипни елементи

//2. масивът има постоянна дължина array.length

//3. дължина на масив = максималния брой елементи, които можем да съхраним

//4. позиции / индекси -> 0 до последната (array.length - 1)

//5. задавате стойност в масив: array[0] = 56;

//6. достъпвате стойност в масив: array[5]

* 1. **Създаване на масив**

double[] prices = new double[10]; - празен

String[] cars = {"Volvo", "BMW", "Ford", "Mazda"}; - запълнен при създаване

* 1. **Пълнене на масив**

//вариант 1 за запълване на масив(статичен с предварително зададени елементи)

int [] dates = {4, 5, 6, 7};

//вариант 2 за запълване на масив (празен масив и добавяме елементи)

double[] prices = new double[10];

prices[0] = 34.5;

prices[1] = 23.5;

//вариант 3 за запълване на масив (**елементите се въвеждат от конзолата на отделни редове**)

int n = Integer.parseInt(scanner.nextLine()); - брой на елементите на масива

int [] numbers = new int[n];

for (int position = 0; position <= numbers.length - 1; position++) {

numbers[position] = Integer.parseInt(scanner.nextLine());

//вариант 4 за запълване на масив (**елементите са на един ред, разделени с интервал**)

String[] inputs = scanner.nextLine().split(" ");

int [] integerNumbers = Arrays.stream(scanner.nextLine().split(" ")).mapToInt(Integer::parseInt).toArray(); //int

double [] decimalNumbers = Аrrays.stream(scanner.nextLine().split(")).mapToDouble(Double::parseDouble).toArray(); //double

String names = "Desi Ivan Georgi Tanya";

String [] namesArray = names.split(" "); //["Desi", "Ivan", "Georgi", "Tanya"] //string

* 1. **Обединяване и преобразуване на елементите на масив**

String names = "Desi Ivan Georgi Tanya";

System.out.println(String.join(" ", namesArray)); //"Desi Ivan Georgi Tanya" // обединяване

System.out.println(Arrays.toString(numbers).replace("[", "").replace("]", "") // обединяване със замяна на разделител

"Desi".toCharArray() -> ['D', 'e', 's', 'i'] // Стринг към символен масив

* 1. **Извеждане на елементите на масив**

//използваме за обхождане for: има значение позицията на елемента

for (int position = 0; position <= daysOfWeek.length - 1; position++) {

System.out.println(daysOfWeek[position]);

}

//използваме за обхождане foreach: няма значение позицията на елемента

["Monday", "Tuesday", "Wednesday", "Thursday","Friday", "Saturday", "Sunday"]

for (String day : daysOfWeek) {

System.out.println(day);

}

//отпечтване в обратен ред

//[10, 20, 30]

//обратен ред: последната позиция към първата

for (int position = numbers.length - 1; position >= 0; position--) {System.out.print(numbers[position] + " ");}

1. **СПИСЪК - LIST**

**1. Размер на списък**

System.out.println(numbers.size()); //размер на списъка

**2. Използване на елемент от списъка**

numbers.get(1); // взима елемент от списъка на посочената позиция

**3. Добавяне на елемент към списъка**

numbers.add(50); //добавя елемента в края на списъка -> {50}

numbers.add(0, 23); //вмъкваме елемент на дадена позиция; изместваме останалите

numbers.set(1, 45); //заменя елемента на дадена позиция с дадения елемент

**4. Премахване на елемент от списъка**

numbers.remove(Integer.valueOf(50)); //премахва първото срещане дадения елемент от списъка

numbers.remove(1); //премахва елемента на дадената позиция

**5. Отпечатваме списък**

**for цикъл -> трябва да работим с позициите**

for (int position = 0; position <= numbers.size() - 1; position++) {

System.out.println(numbers.get(position));

}

**foreach -> трябва да работим само с елементите, без да се интересуваме от позициите**

for (int number : numbers) {

System.out.println(number);

}

**toString**

//numbers = {3, 5, 6, 12,34, 56};

System.out.println(numbers.toString());

//[3, 5, 6, 12, 34, 56]

**String.join -> само за лист от текстове**

List<String> names = new ArrayList<>(Arrays.asList("Ivan", "Georgi", "Pesho"));

System.out.println(String.join(", ", names));

**6. Други методи на list**

System.out.println(numbers.contains(12)); //проверява дали даден елемент е в листа (по стойност)

System.out.println(numbers.isEmpty()); //проверява дали листът е празен

numbers.clear(); //премахва всички елементи в листа

System.out.println(numbers.indexOf(56)); //връща позицията, на която се намира елемента; връща -1 ако няма такъв елемент

size() – number of elements in the List<E> //извежда броя на елементите

add(element) – adds an element to the List<E> //добавя елемент в края

add(index, element) – inserts an element to given position //добавя елемент на определен индекс

remove(element) – removes an element (returns true / false) //премахва даден елемент

remove(index) – removes element at index // премахва елемент на определен индекс

contains(element) – determines whether an element is in the list //проверява дали списъкът съдържа определен елемент

set(index, item) – replaces the element at the given index //заменя елемент на определен индекс

**7. Четене на текст от конзолата**

//34 56 12 45 87

//към масив

int[] numbersArray = Arrays.stream(scanner.nextLine()

.split(" "))

.mapToInt(Integer::parseInt).toArray();

//int към лист

List<Integer> numbersList = Arrays.stream(scanner.nextLine()

.split(" "))

.map(Integer::parseInt)

.collect(Collectors.toList());

//Стринг към лист

List<String> namesList = Arrays.stream(scanner.nextLine().split(" ")).collect(Collectors.toList());

**7. Сортиране**

7.1. ascending order -> нарастващ ред

Collections.sort(numbers);

7.1. descending order -> намаляващ ред

Collections.sort(numbers); //ascending order

Collections.reverse(numbers); //обратен ред

**8. Всички методи на ЛИСТ**

|  |  |
| --- | --- |
| void add(int index, E element) | Използва се за вмъкване на посочения елемент на посочената позиция в списък. |
| boolean add(E e) | Използва се за добавяне на посочения елемент в края на списък. |
| boolean addAll(Collection<? extends E> c) | Използва се за добавяне на всички елементи в указаната колекция в края на списък. |
| boolean addAll (int index, Collection<? extends E> c) | Използва се за добавяне на всички елементи в посочената колекция, започвайки от посочената позиция в списъка. |
| void clear() | Използва се за премахване на всички елементи от този списък. |
|  |  |
| boolean equals(Object o) | Използва се за сравняване на посочения обект с елементите на списък. |
| int hashcode() | Използва се за връщане на стойността на хеш кода за списък. |
| E get(int index) | Използва се за извличане на елемента от определена позиция в списъка. |
| boolean isEmpty() | Връща true, ако списъкът е празен, в противен случай false. |
| int lastIndexOf(Object o) | Използва се за връщане на индекса в този списък на последното срещане на посочения елемент или -1, ако списъкът не съдържа този елемент. |
| Object[] toArray() | Използва се за връщане на масив, съдържащ всички елементи в този списък в правилния ред. |
| <T> T[] toArray(T[] a) | Използва се за връщане на масив, съдържащ всички елементи в този списък в правилния ред. |
| boolean contains(Object o) | Връща true, ако списъкът съдържа посочения елемент |
| boolean containsAll(Collection<?> c) | Връща true, ако списъкът съдържа всички посочени елементи |
| int indexOf(Object o) | Използва се за връщане на индекса в този списък на първото появяване на посочения елемент или -1, ако списъкът не съдържа този елемент. |
| E remove(int index) | Използва се за премахване на елемента, присъстващ на посочената позиция в списъка. |  |
| boolean remove(Object o) | Използва се за премахване на първото появяване на посочения елемент. |  |
| boolean removeAll(Collection<?> c) | Използва се за премахване на всички елементи от списъка. |  |
| void replaceAll(UnaryOperator<E> operator) | Използва се за замяна на всички елементи от списъка с посочения елемент. |  |
| void retainAll(Collection<?> c) | Използва се за запазване на всички елементи в списъка, които присъстват в указаната колекция. |  |
| E set(int index, E element) | Използва се за заместване на посочения елемент в списъка, присъстващ на посочената позиция. |  |
| void sort(Comparator<? super E> c) | Използва се за сортиране на елементите от списъка на базата на зададен компаратор. |  |
| Spliterator<E> spliterator() | Използва се за създаване на сплитератор върху елементите в списък. |  |
| List<E> subList(int fromIndex, int toIndex) | Използва се за извличане на всички елементи в дадения диапазон. |  |
| int size() | Използва се за връщане на броя елементи, присъстващи в списъка. |  |

**9. Сравнение на ЛИСТ и МАСИВ**

**1. Съзадаване**

int [] array = new int[10]; //задаваме брой на елементите

List<Integer> list = new ArrayList<>(); //няма нужда да задаваме брой елементи

**2. Брой елементи**

System.out.println(array.length); //дължина = брой елементи

System.out.println(list.size()); //размер = брой елементи

**3. Достъп елементи**

System.out.println(array[0]);

System.out.println(list.get(0));

**4. Добавяне елементи**

array[1] = 5;

list.add(50); //добавяме елемента в скобите в края на списъка

list.add(0, 12); //вмъквате елемента на дадения индекс

**5. Обхождане с foreach**

for (int number: array) {

System.out.println(number);

}

for (int number: list) {

System.out.println(number);

}

**6. Обхождане с for**

for (int position = 0; position <= array.length - 1; position++) {

System.out.println(array[position]);

}

for (int position = 0; position <= list.size() - 1; position++) {

System.out.println(list.get(position));

}

//ПРЕДИМСТВА НА ЛИСТ ПРЕД МАСИВ

//1. ПРЕОРАЗМЕРЯВАНЕ - НЯМА НУЖДА ПРЕДВАРИТЕЛНО ДА ЗНАЕМ БРОЯ НА ЕЛЕМЕНТИ

//2. ПО-ФУНКЦИОНАЛЕН = ПО-ЛЕСНО СЕ МОДИФИЦИРА

1. **МЕТОДИ**

**Определение:**

Именуван блок от код, който може да бъде използван по-късно

**Дефиниране (деклариране)**

Без параметри

public static void name () {

System.out.println("Hello!");

}

С параметри

static void printNumbers(int start, int end) {

for (int i = start; i <= end; i++) {

System.out.printf("%d ", i);

} }

**Използване**

printHello (); - без параметри

printNumbers (6, 12); - с параметри

**Видове**: връщат стойност и не връщат стойност (void),

**Overloading**: когато няколко метода са с еднакво име, но с различна сигнатура (входни параметри). Всеки метод може да връща различен тип данни.

1. **ОБЕКТИ И КЛАСОВЕ**

public class Book {

**1. Fields – полета**

//характеристики - конски сили, марка, цвят

//private - достъпваме само в рамките на класа

//public - достъпвам навсякъде в класовете в проекта

private String title;

private String author;

private double price;

**2. Constructor – конструктор, носи точно името на класа**

//конструктори - public методи, чрез които създаваме обекти от класа

//1. default constructor -> създава празен обект от класа

//2. custom constructor -> създава обект, на който мога да задам стойности на полетата

public Book (String title, String author, double price) {

this.title = title;

this.author = author;

this.price = price;

}

**3. Methods - действия**

public void sell() {

System.out.printf("Book with title: %s was successfully sold for %.2f.", this.title, this.price);

}

1. **АСОЦИАТИВНИ МАСИВИ - MAP**

**//съзадаване на празен map**

Map<String,Double> studentsMap = new TreeMap<>();

**//видове:**

//1. HashMap -> редът на записите не е гарантиран

//2. LinkedHashMap -> редът на записите се запазва спрямо реда на добавяне

//3. TreeMap -> нарежда записите спрямо ключа в нарастващ ред (ascending order)

**//добавяме записи в map**

studentsMap.put("Ivan", 5.60);

salariesMap.putIfAbsent("Ivan", 3450.50); //добавя ако такъв ключ няма

**// Използване на записи от мапа**

map.getKey() - взима ключа

map.getValue() – взима стойността

**// Брой на записи в мапа**

System.out.println(studentsMap.size());

**//премахваме записи от map**

studentsMap.remove("Petya"); //премахване по ключ, ако го има

studentsMap.remove("Georgi", 3.40); //премахване на запис, ако го има

**//проверка дали map е празен (size = 0)**

System.out.println(studentsMap.isEmpty());

**//проверка дали съществува запис с даден ключ или стойност**

System.out.println(studentsMap.containsKey("Desi"));

System.out.println(studentsMap.containsValue(5.60));

**//премахва всички елементи от map**

studentsMap.clear();

**//Отпечатване**

//1 начин -> StreamAPI

words = ["kiwi", "orange", "banana"]

Arrays.stream(words).forEach(word -> System.out.println(word)); //метод на Arrays.stream

words.entrySet().forEach(entry -> System.out.printf("%s -> %d\n", entry.getKey(), entry.getValue())); // метод на мапа

//2 начин -> foreach

for (String word : words) {

System.out.println(word);

}

**Lambda\_StreamAPI**

API - Application Program Interface

Stream API - съвкупоност от методи върху структури данни

2 вида stream / поточна линия / поток

1. Primitive -> IntStream, DoubleStream -> sum, average, min, max

2. Stream -> Stream<String>, Stream<Integer>

int [] numbers = Arrays.stream(input).mapToInt(e -> Integer.parseInt(e)).toArray(); - парсване на елементите на масив input

int min = Arrays.stream(numbers).min().orElse(23); - добавя в променливата min най-малкия елемент на масива или 23

int[] nums = Arrays.stream(scanner.nextLine().split(" ")).mapToInt(e -> Integer.parseInt(e)).filter(n -> n > 0).toArray(); - филтрира само елементите по – големи от нула и ги добавя към масив

String[] words = Arrays.stream(scanner.nextLine().split(" ")) //["kiwi", "orange", "banana", "apple"]

.filter(word -> word.length() % 2 == 0) //["kiwi", "orange", "banana"] //само думите с четен брой букви

.toArray(String[] :: new); // към масив

1. **ОБРАБОТКА НА ТЕКСТОВЕ**

**String name = "Desislava";**

**1. дължина на текст = брой символи**

System.out.println(name.length());

**2. само главни букви**

System.out.println(name.toUpperCase()); - преобразува стринга към главни букви

**3. само с малки букви**

System.out.println(name.toLowerCase()); преобразува стринга към малки букви

**4. Преобразуване в масив от символи: "Desislava" -> ['D', 'e', 's', 'i', 's', 'l', 'a', 'v', 'a']**

char [] symbols = name.toCharArray();

**5. сравняване на текстове -> true, false**

System.out.println("Desislava".equals(name)); // проверява еднакви ли са текстовете

System.out.println("DeSIslava".equalsIgnoreCase(name)); // проверява еднакви ли са текстовете без да прави разлика между малки и главни букви

**6. достъпване символ от текста**

//позиция (индекс) = 0 -> първия символ

//позиция (индекс) = дължина на текст - 1 -> последния символ

System.out.println(name.charAt(0));

System.out.println(name.charAt(name.length() - 1));

**7. съдържа определен текст -> true (ако се съдържа), false (ако не се съдържа)**

System.out.println(name.contains("Des"));

**8. започва с опеределен текст -> true (ако започва), false (ако не започва)**

System.out.println(name.startsWith("De"));

**9. завършва на определен текст -> true (ако завършва), false (ако не завършва)**

System.out.println(name.endsWith("lava"));

**10. премахва интервалите в началото и края на текста**

System.out.println(" Desislava ".trim());

**11. заменя първото срещане на даден текст / символ (**да се провери, май заменя всички срещания на текста)

name.replace("va", "ta"); //"Desislava" -> "Desislata"

name.replace('v', 'r'); //"Desislava" -> "Desislara"

**12. заменя всички срещания на текста**

name.replaceAll("a", "b"); //"Desislava" -> "Desislbvb"

**13. повтаряме текста даден брой пъти**

System.out.println(name.repeat(5)); //"DesislavaDesislavaDesislavaDesislavaDesislava"

**14. текст, който е част от друг текст**

System.out.println(name.substring(2)); //от посочената позиция до края //"Desislava" -> "sislava"

System.out.println(name.substring(1, 5)); - (Вторият индекс не се включва – до него, но без него. Първият се включва) „Desislava" -> "esis"

**15. Обхождане на текст**

**//обходим един текст от първия към последния символ**

for (int position = 0; position <= name.length() - 1; position++) {

char currentSymbol = name.charAt(position);

System.out.println(currentSymbol);

}

**//обходим един текст от последния към първия символ**

for (int position = name.length() - 1; position >= 0; position--) {

char currentSymbol = name.charAt(position);

System.out.println(currentSymbol);

}

**//конкатенация -> долепяне / събиране на два текста**

//1. чрез оператора +

String result = "Desi" + " " + "Topuzakova";

//2. чрез метода concat

String concatResult = "Ivan".concat(" ").concat("Ivanov");

**//Join - обединяване**

String text = String.join(",", "con", "ca", "ten", "ate");

String[] textsArray = new String[] {"Ivan", "Georgi", "Peter"};

System.out.println(String.join("-", textsArray));

**//Split - разделяне**

String input = "Ivan-Peter-John-George";

String [] words = input.split("-");

**//Substring**

String partInput = input.substring(0, 4); //position: [0;4)

String partInput2 = input.substring(8);//position: [8; length - 1]

**//Searching -> indexOf, lastIndexOf, contains**

String fruits = "banana, apple, kiwi, banana, apple";

System.out.println(fruits.indexOf("banana"));

System.out.println(fruits.lastIndexOf("apple"));

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

System.out.println(fruits.contains("pineapple")); //false

**//повтаряне на текст**

String animal = "turtle";

System.out.println(animal.repeat(5));

**//замяна**

//замяна на всички срещания -> replace

//замяна на първото срещане -> replaceFirst

String lastNamePeople = "Ivanov Ivanov Petrov Georgiev";

lastNamePeople = lastNamePeople.replace("Ivanov", "Petkov");

**//EXAMPLE**

String test = "I am enjoying programming";

System.out.println(test.substring(5, 13));

String text = "enjoying";

int index = test.indexOf(text); //индексът на първата буква на текста = 5

System.out.println(test.substring(index, index + text.length()));

**//STRING BUILDER – използва се винаги, когато има операции върху стрингове – долепяне и др.**

StringBuilder sb = new StringBuilder(); //празен string builder

лесно се модифицира текста, бърз

sb.append("Hello, ");

sb.append("John! ");

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

System.out.println(sb.toString());

StringBuilder sbName = new StringBuilder("Desi");

**//добавяне на текст**

sbName.append(" Topuzakova");

**//използване на текста**

System.out.println(sbName.toString());

String text – sb.toString();

**//изтривне на елементи от текст**

sbName.delete(3, 9);

**//обръщане на текст**

String fullName = "Desislava Topuzakova";

StringBuilder fullNameBuilder = new StringBuilder(fullName);

fullNameBuilder.reverse();

System.out.println(fullNameBuilder);

**//Дължина на текст**

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

**//вмъкване в текст**

sb.insert(0, "Ivan");

**//достъпване на символ в текста в StringBuilder**

System.out.println(sb.charAt(0));

1. **REGULAR EXPRESSIONS - REGEX**

**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)

[A-Za-z] - една буква, която или е малка, или е голяма

[aeiou] - всички гласни букви

[^aeiou] - всички съгласни букви

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

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

\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

() -> обособяваме група. Името е автоматично, започва от 1 ......

(?<name>) -> обособяваме група с име

\b -> слагаме граница, която казва, че не искаме да има символи(букви/цифри) преди/след съвпадението, което е открито в текста.

**Брой на срещанията:**

\* -> срещания 0 или безброй много пъти

+ -> срещания 1 или безброй много пъти

? -> срещания 0 или 1 пъти

{число} -> срещания {число} пъти

{число, } -> минимум колко пъти

{число1, число2} -> минимум се среща число1 пъти, максимум се среща число2 брой пъти

() -> обособяваме група

(?<name> шаблон) -> обособяваме група с име

**Използване в Java:**

//Прочитане на текста от конзолата

String text = scanner.nextLine();

// 1. Създаване на текст на шаблона (критерии за търсене, шаблон. Проверява се в сайта <https://regex101.com/>

String regex = "\\+359([ -])2\\1\\d{3}\\1\\d{4}\\b";

// 2. Създаване на Java шаблон

Pattern pattern = Pattern.compile(regex);

// 3. Създаване на инструмент с помощта на който ще проверяваме за съвпадения в прочетеният текст от конзолата. Използва се класа Мatcher.

Matcher matcher = pattern.matcher(text);

// 4. Създаване на Списък за запазване на откритите съвпадения

List<String> validNumbers = new ArrayList<>();

//5. Проверяване за съвпадения в текста, използват се методи на класа Мatcher

* find – връща true ако намери съвпадение и обратно
* group – взима намереното съвпадение

while (matcher.find()){

validNumbers.add(matcher.group())}

System.out.println(String.join(", ", validNumbers))}

//6. Използване на части от съвпаденията

* разделяне на регекса на групи

String regex = "\\b(?<day>\\d{2})([\\.\\-\\/])(?<month>[A-Z][a-z]{2})\\2(?<year>\\d{4})\\b"; //13/Jul/1928

* използване на отделните части на съвпаденията

String wholeMatch = matcher.group(); //"13/Jul/1928"

String day = matcher.group("day"); или String day = matcher.group(1); //"13"

String month = matcher.group("month"); или String month = matcher.group(3); //"Jul"

String year = matcher.group("year"); или String year = matcher.group(4); //"1928"

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ЛИНЕЙНИ СТРУКТУРИ ОТ ДАННИ - ОБОБЩЕНИЕ** | | | | | | |
|  | **Масив** | **Списък** | **Асоциативни масиви - MAP** |  | **Текстови променливи** | **STRING BUILDER** |
| **Видове** |  |  | 1. HashMap -> редът на записите не е гарантиран  2. LinkedHashMap -> редът на записите се запазва спрямо реда на добавяне  3. TreeMap -> нарежда записите спрямо ключа в нарастващ ред (ascending order) |  |  |  |
| **Създаване** | int [] array = new int[10];  //задаваме брой на елементите | List<Integer> list = new ArrayList<>();  //няма нужда да задаваме брой елементи | Map<String,Double> studentsMap = new TreeMap<>(); | **Създаване** | String string = “ ” | StringBuilder sb = new StringBuilder(); |
| **Брой елементи** | array.length(); | list.size(); | Map.size(); | **Брой елементи** | String.length(); | sb.length(); |
| **Използване на елементи** | System.out.println(array[0]); | System.out.println(list.get(0)); | map.getKey() - взима ключа  map.getValue() – взима стойността | **Използване на елементи** | System.out.println(string.charAt(0));  //връща символа на посочената позиция | System.out.println(sbName.toString());  //преобразува към стринг  System.out.println(sb.charAt(0));  //връща символа на определена позиция |
| **Добавяне елементи** | array[1] = 5; | list.add(50);  //добавяме елемента в скобите в края на списъка  list.add(0, 12);  //вмъквате елемента на дадения индекс | Map.put("Ivan", 5.60);  Map.putIfAbsent("Ivan", 3450.50);  //добавя ако такъв ключ няма | **Само главни букви** | System.out.println(name.toUpperCase());  - преобразува стринга към главни букви | sb. toUpperCase()  - преобразува стринга към главни букви |
| **Премахване на елементи** |  | **remove** (int index) | Map.remove("Petya"); //премахване по ключ, ако го има  Map.remove("Georgi", 3.40); //премахване на запис, ако го има  studentsMap.clear();  //премахва всички елементи | **Само малки букви** | System.out.println(name.toLowerCase());  - преобразува стринга към малки букви | sb. toLowerCase()  - преобразува стринга към малки букви |
| **Обхождане с foreach** | for (int number: array) {  System.out.println(number);  } | for (int number: list) {  System.out.println(number);  } | for (String word : words) {  System.out.println(word);  } | **Обхождане от първия към последния символ** | for (int position = 0; position <= name.length() - 1; position++) {  char currentSymbol = name.charAt(position);  System.out.println(currentSymbol);  } |  |
| **Обхождане с for** | for (int position = 0; position <= array.length - 1; position++) {  System.out.println(array[position]);  } | for (int position = 0; position <= list.size() - 1; position++) {  System.out.println(list.get(position));  } |  | **Обхождане от последния към първия символ** | for (int position = name.length() - 1; position >= 0; position--) {  char currentSymbol = name.charAt(position);  System.out.println(currentSymbol);  } |  |
| **Обхождане със StreamAPI** |  |  | Arrays.stream(words).forEach(word -> System.out.println(word)); //метод на Arrays.stream  words.entrySet().forEach(entry -> System.out.printf("%s -> %d\n", entry.getKey(), entry.getValue())); // метод на мапа | **Добавяне**  **Конкатенация -> долепяне** | //1. чрез оператора +  String result = "Desi" + " " + "Topuzakova";  //2. чрез метода concat  String concatResult = "Ivan".concat(" ").concat("Ivanov"); | sbName.append(" Topuzakova");  //добавяне накрая  sb.insert(0, "Ivan");  //вмъкване на посочената позиция |
| **Проверка дали е празен** |  | list.size() ==0; | Map.isEmpty() | **Join - обединяване** | String text = String.join(",", "con", "ca", "ten", "ate");  String[] textsArray = new String[] {"Ivan", "Georgi", "Peter"};  System.out.println(String.join("-", textsArray)); | След прехвърляне към String с метода toString() |
| **Методи** |  |  |  | **Split - разделяне** | String input = "Ivan-Peter-John-George";  String [] words = input.split("-"); | След прехвърляне към String с метода toString() |
| Вмъкване на посочения елемент на посочената позиция в списък. |  | void **add**(int index, E element) |  | **Substring** | String partInput = input.substring(0, 4);  Между посочените позиции  String partInput2 = input.substring(8)  От посочената позиция до края | sb.substring(5)  взима подстринг, започвайки от посочената позиция  sb.substring(5, 8)  между посочените позиции |
| Премахване на всички елементи от този списък. |  | void **clear**() |  | **Преобразуване в масив от символи:** | char [] symbols = name.toCharArray(); | След прехвърляне към String с метода toString() |
| Извличане на елемента от определена позиция в списъка. |  | E **get** (int index) | keySet()  -> връща всички ключове от всички записи  entrySet()  -> връща колекция от всички записи  get(ключ)  -> връща стойността, която стои срещу дадения ключ | **Сравняване на текстове -> true, false** | System.out.println("Desislava".equals(name)); проверява еднакви ли са текстовете  System.out.println("DeSIslava".equalsIgnoreCase(name)); проверява еднакви ли са текстовете без да прави разлика между малки и главни букви | System.out.println(sb1.equals(sb2));  System.out.println(sb1.toString() == sb2.toString())  проверява еднакви ли са текстовете |
| Връща true, ако списъкът е празен, в противен случай false. |  | boolean **isEmpty**() |  | **Дали съдържа определен текст** | System.out.println(name.contains("Des"));  -> true (ако се съдържа), false (ако не се съдържа) |  |
| Връща true, ако списъкът съдържа посочения елемент |  | boolean **contains**(Object o) | containsKey(key)  -> проверява дали в map-а има запис с такъв ключ -> резултат true ако има, false ако няма  containsValue(value)  -> проверява дали в map-а има запис с такова value -> резултат true ако има, false ако няма | **Дали започва с опеределен текст** | System.out.println(name.startsWith("De"));  -> true (ако започва), false (ако не започва) |  |
| Премахване на елемента, присъстващ на посочената позиция в списъка. |  | E **remove**(int index) |  | **Дали завършва на определен текст** | System.out.println(name.endsWith("lava"));  -> true (ако завършва), false (ако не завършва) |  |
| Замяна на всички елементи от списъка с посочения елемент. |  | void **replaceAll**(UnaryOperator<E> operator) |  | **Премахване на интервалите в началото и края на текста** | System.out.println(" Desislava ".trim()); |  |
| Заместване на посочения елемент в списъка, присъстващ на посочената позиция. |  | E **set**(int index, E element) |  | **Заменя първото срещане на даден текст** | name.replace('v', 'r'); //"Desislava" -> "Desislara" | StringBuilder replace(int start, int end, String str)  Заменя частта от низа, указана от интервала start..end, с подаден низ |
| Връщане на броя елементи, присъстващи в списъка. |  | int **size**() |  | **Заменя всички срещания на текста** | name.replaceAll("a", "b"); //"Desislava" -> "Desislbvb" |
| Връща позицията, на която се намира елемента; връща -1 ако няма такъв елемент |  | list.**indexOf**(56)); |  | **Повтаряне на текста даден брой пъти** | System.out.println(name.repeat(5)); //"DesislavaDesislavaDesislavaDesislavaDesislava" |  |
| Добавяне на посочения елемент в края на списък. |  | boolean add(E e) |  | **Текст, който е част от друг текст** | System.out.println(name.substring(2)); //от посочената позиция до края //"Desislava" -> "sislava"  System.out.println(name.substring(1, 5)); - (Вторият индекс не се включва – до него, но без него. Първият се включва) „Desislava" -> "esis" | String substring (int start, int end)  Връща подниза, дефиниран от посочения интервал |
| Добавяне на всички елементи в указаната колекция в края на списък. |  | boolean addAll(Collection<? extends E> c) |  | **Търсене -> indexOf, lastIndexOf, contains** | String fruits = "banana, apple, kiwi, banana, apple";  System.out.println(fruits.indexOf("banana"));  Връща първия индекс на намерено съвпадение  System.out.println(fruits.lastIndexOf("apple"));  Връща последния индекс на намерено съвпадение  System.out.println(fruits.contains("kiwi"));  Връща true, ако има намерено съвпадение | int indexOf(String str, int index)  Търси подниз в текущия низ  int lastIndexOf(String str, int index)  Търси подниз в текущия низ, започвайки от края |
| Добавяне на всички елементи в посочената колекция, започвайки от посочената позиция в списъка. |  | boolean addAll (int index, Collection<? extends E> c) |  | **Изтриване** |  | sbName.delete(3, 9);  Изтрива символите между посочените индекси |
| Сравняване на посочения обект с елементите на списък. |  | boolean equals(Object o) |  | **Обръщане на текст** |  | StringBuilder reverse()  Обръща текущия низ. |
| Връща стойността на хеш кода за списък. |  | int hashcode() |  |  |  |  |
| Връща индекса в този списък на последното срещане на посочения елемент или -1, ако списъкът не съдържа този елемент. |  | int lastIndexOf(Object o) |  |  |  |  |
| Връща масив, съдържащ всички елементи в този списък в правилния ред. |  | Object[] toArray() |  |  |  |  |
| Връща масив, съдържащ всички елементи в този списък в правилния ред. |  | <T> T[] toArray(T[] a) |  |  |  |  |
| Връща true, ако списъкът съдържа всички посочени елементи |  | boolean containsAll(Collection<?> c) |  |  |  |  |
| Връща индекса в този списък на първото появяване на посочения елемент или -1, ако списъкът не съдържа този елемент. |  | int indexOf(Object o) |  |  |  |  |
| Премахване на първото появяване на посочения елемент. |  | boolean remove(Object o) |  |  |  |  |
| Премахване на всички елементи от списъка. |  | boolean removeAll(Collection<?> c) |  |  |  |  |
| Запазване на всички елементи в списъка, които присъстват в указаната колекция. |  | void retainAll(Collection<?> c) |  |  |  |  |
| Сортиране на елементите от списъка на базата на зададен компаратор. |  | void sort(Comparator<? super E> c) |  |  |  |  |
| Създаване на сплитератор върху елементите в списък. |  | Spliterator<E> spliterator() |  |  |  |  |
| Извличане на всички елементи в дадения диапазон. |  | List<E> subList(int fromIndex, int toIndex) |  |  |  |  |