CLASA String

Metode şi exemple

Crearea unui şir

- Constructorul implicit crează un şir vid:

  String s = new String();

- String str = "abc";

este echivalent cu:

  char data[] = {'a', 'b', 'c'};

  String str = new String(data);

OBS. Dacă elementele din vectorul data sunt modificate după crearea şirului str, atunci aceste modificări nu apar în acest şir.

- Construirea unui şir pe baza altui şir:

  String str2 = new String(str);

Operații cu şiruri

- Metoda **length()** returnează lungimea unui şir

  Ex: System.out.println(“Hello”.length()); // afişează 5

- Caracterele dintr-un şir pot fi accesate astfel:

  public char **charAt**(int index);

  Returnează caracterul din poziția index. Domeniul de indexare este de la 0 la length() - 1.

  Ex. char ch;

  ch = “abc”.charAt(1); // ch = “b”

Operații cu şiruri

- Concatenarea: +

  String s1 = "ab" + "cd";

  String s2 = s1 + 123 + "xyz“
- extrem de flexibil, permite concatenarea şirurilor cu obiecte de orice tip care au o reprezentare de tip şir de caractere.

- Exemple:

System.out.print("Vectorul v are" + v.length + "elem.");

String x = "a" + 1 + "b“;

- De fapt:

String x = new StringBuffer().append("a").append(1).

append("b").toString()

- Obs: şirul s=1+2+"a"+1+2 va avea valoarea "3a12", primul + fiind operatorul matematic de adunare iar al doilea +, cel de concatenare a şirurilor.

Operații cu şiruri

- **getChars() -** Copiază caracterele din şirul sursă în şirul destinație

  public void **getChars**(int srcBegin, int srcEnd, char[] dst, int dstBegin)

  - srcBegin – indexul primului caracter din sursă
  - srcEnd – indexul ultimului caracter din sursă
  - dst – vectorul destinație
  - dstBegin – poziția de la care începe copierea în vectorul destinație
- **equals() –** Compară două şiruri la egalitate

  public boolean **equals**(String s2)
- **equalsIgnoreCase()-** Compară două şiruri la egalitate fără să țină cont de litere mici sau mari

  public boolean **equalsIgnoreCase**(String s2)

Operații cu şiruri

- **startsWith()** – Testează dacă şirul începe cu prefixul specificat (la început sau dintr-o anumită poziție)

  public boolean **startsWith**(String prefix)

  Ex.“Figure”.startsWith(“Fig”); // true

  public boolean **startsWith**(String prefix, int toffset)

  Ex.“figure”.startsWith(“gure”, 2); // true

- **endsWith() -** Testează dacă şirul se termină cu sufixul specificat

  public boolean **endsWith**(String suffix)

  Ex. “Figure”.endsWith(“re”); // true

- **compareTo()** – Compară două şiruri din punct de vedere lexicografic

  Rezultatul este

  - Negativ, dacă şirul precede şirul primit ca argument
  - Zero, dacă şirurile sunt egale
  - Pozitiv, dacă şirul urmează şirului primit ca argument

  public int **compareTo**(String anotherString)

  public int **compareToIgnoreCase**(String str)

Operații cu şiruri

- **indexOf –** Caută prima apariție a unui caracter sau şir într-un alt şir. Returnează -1 dacă nu îl găseşte.

  public int **indexOf**(int ch)

  public int **indexOf**(String str)

  Ex. String str = “How was your day today?”;
  str.indexOf(‘o’); // 1
  str.indexOf(“was”); //4

- căutare începând de la o poziție specificată:
  public int **indexOf**(int ch, int fromIndex)
  public int **indexOf**(String str, int fromIndex)

  Ex. String str = “How was your day today?”;
  str.indexOf(‘a’, 6); //14
  str.indexOf(“was”, 2); //4

- **lastIndexOf()** - Caută ultima apariție a unui caracter sau şir într-un alt şir, similar lui indexOf.

Operații cu şiruri

- **substring()** – Returnează un nou şir care este un subşir al şirului sursă.

  public String **substring**(int beginIndex)

  Ex: "unhappy".substring(2); // returnează "happy"

  public String **substring**(int beginIndex,
  int endIndex)

  Ex: "smiles".substring(1, 5); // returnează "mile“
- **concat()** – Concatenează şirul specificat la şirul sursă

  public String **concat**(String str)

  "to".concat("get").concat("her"); // returnează "together"
- **replace()-** Returnează un nou şir în care toate aparițiile caracterului oldChar sunt înlocuite cu caracterul newChar.

  public String **replace**(char oldChar, char newChar)

  Ex. "mesquite in your cellar".replace('e', 'o'); returnează "mosquito in your collar"

Operații cu şiruri

- **trim() -** Returnează şirul fără spațiile albe de la început şi sfârşit

  public String trim()

  Ex. String s = “ Hi Mom! “.trim();

  S este acum “Hi Mom!”

- **valueOf() –** Returnează reprezentarea sub formă de şir de caractere a argumentului

  public static String **valueOf**(char[] data)

  public static String **valueOf**(char c)

  public static String **valueOf**(boolean b)

  public static String **valueOf**(int i)

  public static String **valueOf**(long l)

  public static String **valueOf**(float f)

  public static String **valueOf**(double d)

Operații cu şiruri

- **toLowerCase():** Converteşte toate caracterele la litere mici

  public String **toLowerCase**()

- **toUpperCase():** Converteşte toate caracterele la litere mari

  public String **toUpperCase**()

  Ex: “HELLO THERE”.toLowerCase();

  “hello there”.toUpperCase();

Operații cu șiruri

public String[] split(String regex)

- împarte sirul in subsiruri folosind un sir de delimitatori (expresie regulata)
- regex – expresia regulata de delimitare
- returneaza un vector de String-uri

Exemplu: pentru sirul "boo:and:foo“ avem urmatoarele rezultate:

regex	rezultat
:	{ "boo", "and", "foo" }
o	{ "b", "", ":and:f" }

public String[] split(String regex, int limit)

- Limit: de cate ori se aplica expresia regulata

StringBuffer

- Este asemănător unui obiect String doar că este modificabil în sensul că există metode pentru modificarea lungimii şi a conținutului său – append, insert.
- Crearea unui obiectStringBuffer:
  - StringBuffer()
  - StringBuffer(int size)
  - StringBuffer(String str)
- Principalele operații sunt adăugarea, inserarea şi ştergerea.
- Adăugarea:
  StringBuffer append(String str)
  StringBuffer append(int num)
- Inserarea se face într-o anumită poziție.
  StringBuffer insert(int index, String str)
  StringBuffer append(int index, char ch)

Operații cu StringBuffer

- **delete()** - şterge un subşir. Subşirul începe în poziția specificată şi se termină în poziția index de final - 1 sau până la sfârşitul şirului dacă nu se specifică poziție de final.

public StringBuffer **delete**(int start, int end)

public StringBuffer **delete**(int start)

- **replace()** – Înlocuieşte un subşir din şirul sursă cu un alt şir

  public StringBuffer **replace**(int start, int end, String str)

- **substring()**

  public String substring(int start)

- **reverse()** – reversul şirului (ordine inversă a caracterelor)

  public StringBuffer **reverse**()

- **length()**

  public int **length**()

Operații cu StringBuffer

- **capacity() –** Returnează capacitatea curentă a StringBuffer-ului. Capacitatea este dimensiunea disponibilă pentru adăugarea de noi elemente

  public int **capacity**()
- **charAt()**

  public char **charAt**(int index)
- **getChars()**

  public void **getChars**(int srcBegin, int srcEnd, char[] dst, int dstBegin)
- **setLength**() **–** Setează lungimea pentru obiectul de tip StringBuffer.

  public void **setLength**(int newLength)

Exemple: StringBuffer

StringBuffer sb = new StringBuffer(“Hello”);

sb.length(); // 5

sb.capacity(); // 21 (16 caractere sunt adăugate dacă nu se specifică o dimensiune)

sb.charAt(1); // e

sb.setCharAt(1,’i’); // Hillo

sb.setLength(2); // Hi

sb.append(“l”).append(“l”); // Hill

sb.insert(0, “Big “); // Big Hill

sb.replace(3, 11, “”); // Big

sb.reverse(); // giB

Clasa StringTokenizer

- Utilizată pentru împărțirea în atomi lexicali (tokens):

public **StringTokenizer**(String str, String delim)

public boolean **hasMoreTokens**()

public String **nextToken**()

Exemplu:

```
StringTokenizer st = new StringTokenizer("this, is a
    test.“,” ,.”);
while (st.hasMoreTokens()) {
    System.out.println(st.nextToken());
}
```

Are următorul rezultat:

this

is

a

test