

Programare orientată pe obiecte

#10 JAVA Clasa String în Java

<https://www.runceanu.ro/adrian/activitate-didactica>

Curs 10

Limbajul JAVA. Clasa String în Java



Clasa String in Java

1. **Siruri de caractere (String-uri)**
2. Concatenarea sirurilor de caractere
3. Alte metode din clasa String
4. Conversii intre siruri de caractere (String) si alte tipuri de date primitive

1. Șiruri de caractere (String-uri)

- Șirurile de caractere în **Java** sunt definite folosind clasa **String**.
- Limbajul **Java** face să pară ca **String** este un tip primitiv, deoarece pentru el sunt definiți operatorii **+** și **+=** pentru concatenare, deși, știm, ca operatorii, în general, nu pot fi aplicați obiectelor.
- Totuși, **String** este *singurul tip referință pentru care **Java** a permis supraincercarea operatorilor.*

1. Șiruri de caractere (String-uri)

Reguli fundamentale privind obiectele de tip **String**:

1. Obiectele de tip **String** se comporta ca orice alt obiect Java, exceptand faptul ca asupra lor se poate aplica operatorul de concatenare;
2. Obiectele de tip **String** sunt nemodificabile (sunt constante), in sensul ca, daca doua variabile referinta indica acelasi sir de caractere, modificarea valorii sirului de caractere catre care refera una din variabile nu va avea nici un efect asupra valorii sirului de caractere pe care o refera cealalta variabila.

1. Șiruri de caractere (String-uri)

- Aceasta inseamna ca, odata construit un obiect de tip **String** nu mai poate fi modificat.
- Din acest motiv operatorul de atribuire se foloseste pentru **String**-uri la fel ca pentru un tip de data primitiv, adica creaza un nou **String**.

De exemplu:

```
String mesaj1 = "Java";
```

```
String mesaj2 = "Eclipse";
```

```
String mesaj2_repet = mesaj2;
```

1. Șiruri de caractere (String-uri)

- După aceste atribuiri există două obiecte de tip **String**:
 - șirul "Java", referit de variabila `mesaj1` și
 - șirul "Eclipse" referit de variabilele `mesaj 2` și `mesaj2_repet`.
- Deoarece *string*-urile sunt obiecte care nu pot fi modificate, singura posibilitate de a modifica valoarea către care referă variabila `mesaj2_repet` este aceea de a construi un nou obiect de tip **String** și de a-l atribui variabilei `mesaj2_repet`.
- Această operație nu va avea nici un efect asupra valorii pe care o referă `mesaj2`.

1. Șiruri de caractere (String-uri)

Crearea de noi obiecte de tip String folosind constructorii clasei String

- **Clasa String** se afla in pachetul *java.lang* si are definiti mai multi *constructori* pentru crearea si initializarea de **obiecte de tip String**.
- In continuare se vor descrie cativa dintre **constructorii clasei String**, mai des folositi.

1. Șiruri de caractere (String-uri)

1. Constructor utilizat pentru **alocarea unui nou String** care **contine o secventa de caractere stocata intr-un tablou unidimensional de caractere.**

Antetul constructorului este:

```
public String(char[] <valoare>)
```

unde:

- **<valoare>** - tabloul de caractere care reprezinta sursa secventei de caractere a noului **String**.

1. Şiruri de caractere (String-uri)

De retinut este faptul ca, modificarea tabloului de caractere nu afecteaza noul sir de caractere creat astfel.

De exemplu:

```
char[] caractere1 = {'a', 'b', 'c', 'd', 'e', 'f'};  
String sir1 = new String(caractere1);
```

1. Şiruri de caractere (String-uri)

2. Constructor **utilizat pentru alocarea unui nou `String` care contine o secventa de caractere stocata intr-o portiune a unui tablou unidimensional de caractere.**

Antetul constructorului este:

```
public String(char[] <valoare>,  
int <deplasament>,  
int <lungimeSecventa>)
```

1. Șiruri de caractere (String-uri)

unde:

- **<valoare>** - tabloul de caractere care reprezintă sursa secvenței de caractere a noului **String**;
- **<deplasament>** - indexul primului caracter din tabloul de caractere de la care se va începe inițializarea noului **String**;
- **<lungimeSecventa>** - lungimea secvenței de caractere preluată din tabloul de caractere pentru inițializarea noului **String**.

1. Șiruri de caractere (String-uri)

De retinut este faptul ca, modificarea subtabloului de caractere preluat nu afecteaza noul sir de caractere creat astfel.

De exemplu:

```
char[] caractere1 = {'a', 'b', 'c', 'd', 'e', 'f'};  
String sir1 = new String(caractere1, 0, 3);
```

1. Şiruri de caractere (String-uri)

3. Constructor utilizat pentru **alocarea unui nou String** care **contine aceeaşi secvenţă de caractere care este stocată într-un alt şir.**

Cu alte cuvinte, *noul şir creat este o copie a unui alt şir.*

De regula, se foloseşte pentru **a crea duplicatul unui şir.**

Antetul constructorului este:

```
public String(String <şirOriginal>)
```

1. Şiruri de caractere (String-uri)

De exemplu:

```
char[] caractere1 = {'a', 'b', 'c', 'd', 'e', 'f'};
```

```
String sir1 = new String(caractere1);
```

```
String sir2 = new String(sir1);
```

Clasa String in Java

1. Siruri de caractere (String-uri)
2. Concatenarea sirurilor de caractere
3. Alte metode din clasa String
4. Conversii intre siruri de caractere (String) si alte tipuri de date primitive

2. Concatenarea șirurilor de caractere

- Atunci când cel puțin unul dintre operanzi este de tip **String**, operatorul plus (+) realizează concatenarea.
- Rezultatul este o referință către un obiect nou construit de tip **String**.

Exemple:

```
1. sir1_concatenat = "x" + "y" + "z";
```

```
// sir1_concatenat are valoarea "xyz"
```

```
2. sir2_concatenat = "Mihai are " + 11 + " ani";
```

```
/* sir2_concatenat are valoarea "Mihai are 11 ani", numărul  
11 este convertit la String */
```

2. Concatenarea șirurilor de caractere

Operatorul `+=` este, de asemenea, folosit pentru concatenarea șirurilor.

Efectul instrucțiunii:

```
sir1 += sir2;
```

este același cu

```
sir1 = sir1 + sir2;
```

Deci, *sir1* va referi un nou obiect de tip **String** a cărui valoare este *sir1 + sir2*.

2. Concatenarea șirurilor de caractere

Metoda concat poate fi folosită, de asemenea, pentru concatenarea a două șiruri.

Antetul metodei este:

```
public String concat(String <șirConcat>)
```

unde:

- **<șirConcat>** - șirul care se concatenează la sfârșitul șirului curent.
- Metoda **concat** *returnează un nou șir de caractere care este format din șirul de caractere curent urmat de șirul de caractere specificat în parametrul <șirConcat>.*
- Dacă lungimea șirului specificat în parametru **<șirConcat>** este 0, atunci se returnează șirul curent.

2. Concatenarea șirurilor de caractere

De exemplu:

```
String nume = "Popescu";
```

```
String prenume = "Vasile";
```

```
String numePrenume = nume.concat(prenume);
```

```
// returneaza "Popescu Vasile"
```

sau:

```
String sirNou = "la ".concat("Popescu").concat(" Vasile");
```

```
// returneaza "la Popescu Vasile"
```

Java String Concat Example

<http://www.java-examples.com/>

```
1.  /*
2.     Java String Concat Example.
3.     This Java String concat example shows how to concat String in Java.
4.  */
5.  public class JavaStringConcat {
6.      public static void main(String args[]){
7.          /*
8.           * String concatenation can be done in several ways in Java.
9.          */
10.
11.         String str1 = "Hello";
12.         String str2 = " World";
13.
14.         //1. Using + operator
15.         String str3 = str1 + str2;
16.         System.out.println("String concat using + operator : " + str3);
17.
18.         /*
19.          * Internally str1 + str 2 statement would be executed as,
20.          * new StringBuffer().append(str1).append(str2)
21.          *
22.          * String concatenation using + operator is not recommended for large
23.          number
24.          * of concatenation as the performance is not good.
25.          */
26.         //2. Using String.concat() method
27.         String str4 = str1.concat(str2);
28.         System.out.println("String concat using String concat method : " + str4);
29.
30.         //3. Using StringBuffer.append method
31.         String str5 = new StringBuffer().append(str1).append(str2).toString();
32.
33.         System.out.println("String concat using StringBuffer append method : " +
34. str5);
35.     }
```

```
38. Output of Java String concat example would be
39. String concat using + operator : Hello World
40. String concat using String concat method : Hello World
41. String concat using StringBuffer append method : Hello World
42. */
```

2. Concatenarea șirurilor de caractere

Compararea șirurilor de caractere

- Operatorii relationali (<, <=, >, >=) nu funcționează pentru obiecte de tip **String**.
- De asemenea, operatorii de egalitate și inegalitate (==, !=) pentru șiruri de caractere au semnificația precizată la obiecte de tip referință, adică compararea adreselor (referințe către obiecte de tip **String**) și nu valorile conținute de obiectele de tip **String**.

2. Concatenarea șirurilor de caractere

A. Pentru a testa egalitatea (identitatea) a doua obiecte de tip **String**, se folosește metoda **equals**.

Antetul metodei este:

```
public boolean equals(Object <unObiect>)
```

unde:

- **<unObiect>** - reprezintă un obiect de tip **String** cu a cărui valoare se compară șirul curent.

2. Concatenarea șirurilor de caractere

- **Metoda equals** returnează valoarea *true* dacă și numai dacă parametrul *unObiect* nu este *null* și este un obiect de tip **String** care are ca valoare aceeași secvență de caractere ca și a șirului curent.
- De exemplu, următoarea secvență de cod (EqualsSiruri.java) arată modul de apel al metodei **equals** pentru compararea conținutului a două șiruri de caractere, introduse de la tastatură:

```
import java.io.*;
public class EqualsSiruri
{
public static void main(String[] args) throws
    IOException {

char[ ] caractere1 = new char[20];
char[ ] caractere2 = new char[20];
for (int i=0; i <=19; i++)
    caractere1[i] = ' ';
for ( int i=0; i <=19; i++)
    caractere2[i] = ' ';
```

```
BufferedReader br = new BufferedReader(new
    InputStreamReader(System.in));
System.out.println ("Introduceti primul sir");
br.read(caractere1, 0, 20);
String sir1 = new String(caractere1);
System.out.println ("Introduceti al doilea sir");
br.read(caractere2, 0, 20);
String sir2 = new String(caractere2);
boolean rezultat = false;
rezultat = sir1.equals(sir2);
if (rezultat ==true)
    System.out.println ("siruri egale");
else
    System.out.println ("siruri diferite");}
}
```

```

curs10_1.java
3
4 public class curs10_1 {
5
6     public static void main(String[] args) throws IOException {
7         char[] caractere1 = new char[20];
8         char[] caractere2 = new char[20];
9         for (int i=0; i <=19; i++)
10            caractere1[i] = ' ';
11         for ( int i=0; i <=19; i++)
12            caractere2[i] = ' ';
13         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
14         System.out.println ("Introduceti primul sir");
15         br.read(caractere1, 0, 20);
16         String sir1 = new String(caractere1);
17         System.out.println ("Introduceti al doilea sir");
18         br.read(caractere2, 0, 20);
19         String sir2 = new String(caractere2);
20         boolean rezultat = false;
21         rezultat = sir1.equals(sir2);
22         if (rezultat ==true)
23             System.out.println ("siruri egale");
24         else
25             System.out.println ("siruri diferite");
26     }
27 }

```

Problems @ Javadoc Declaration Console

<terminated> curs10_1 [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (11 nov. 2014, 15:18:54)

```

Introduceti primul sir
programare in Java
Introduceti al doilea sir
programare in Java
siruri egale

```

Problems @ Javadoc Declaration Console

<terminated> curs10_1 [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (11 nov. 2014, 15:18:03)

```

Introduceti primul sir
informatica
Introduceti al doilea sir
programare orientata pe obiecte
siruri diferite

```

2. Concatenarea șirurilor de caractere

Nota:

Metoda **read** din clasa *BufferedReader*, definita in pachetul *java.io.Reader*.

Antetul metodei **read** este:

```
public int read(char[ ] <cbuf>, int <deplasament>, int <lungime>)
```

unde:

- **<cbuf>** - tablou de caractere unde se face stocarea secventei de caractere citita din buffer-ul de intrare;
- **<deplasament>** - indexul din tabloul de caractere **<cbuf>** de la care va incepe stocarea;
- **<lungime>** - numarul maxim de caractere care se doresc a fi citite si depuse in tabloul de caractere dat de parametrul **<cbuf>**.

2. Concatenarea șirurilor de caractere

- **Metoda read** citește un număr de caractere, dat de parametrul **<lungime>**, din buffer-ul de intrare și stochează aceste caractere citite într-o porțiune a unui tablou unidimensional de caractere dat de parametrul **<cbuf>**.
- Atunci când se folosește metoda **read** pentru citirea unui buffer de intrare de la tastatură (**System.in**) *este indicat ca valoarea din parametrul <lungime> să fie mai mare decât lungimea secvenței de caractere citite de la tastatură.*

2. Concatenarea șirurilor de caractere

- Altfel, se realizează trunchierea secvenței citite sau trebuie reapelată metoda **read** pentru a prelua și restul caracterelor citite din bufferul de intrare.
- Dacă nu se dorește reapelarea metodei **read** pentru citirea întregului buffer de intrare, atunci pentru citirea unui nou buffer de intrare (a unor alte date de la tastatură) se creează un nou obiect de tip *BufferedReader* care să preia noile date citite de la tastatură.

2. Concatenarea șirurilor de caractere

- De reținut ca, la citirea de la tastatura prin apelul metodei **read** sunt preluate in tabloul de caractere dat de parametrul **<cbuf>** si caracterele:
 - **'\r'** (cod ASCII - 13)
 - si **'\n'** (cod ASCII - 10).
- In functie de necesitati uneori este necesar ca aceste 2 caractere sa fie eliminate din tabloul de caractere **<cbuf>**.

2. Concatenarea șirurilor de caractere

Metoda equalsIgnoreCase este folosită pentru a testa **egalitatea a doua obiecte** de tip **String** *fara a face distincție între literele mici și literele mari* ale alfabetului.

Antetul metodei este:

```
public boolean equalsIgnoreCase(String <altSir>)
```

unde:

- **<altSir>** - un alt obiect de tip **String** cu care se face comparația șirului curent.

2. Concatenarea șirurilor de caractere

Metoda equalsIgnoreCase returnează valoarea *true* dacă parametrul `<altSir>` nu este *null* și dacă cele două șiruri de caractere au aceeași lungime și sunt formate din aceeași secvență de caractere, fără a se face distincție între literele mari și mici ale alfabetului.

De exemplu:

```
boolean rezultat = false;  
rezultat = sir1.equalsIgnoreCase(sir2);
```

2. Concatenarea șirurilor de caractere

B. Metoda compareTo este folosită pentru a realiza un test mai general între două șiruri din punct de vedere al ordinii lexicografice.

Compararea a două șiruri de caractere se bazează pe valoarea Unicode a fiecărui caracter din șirurile de caractere.

Antetul metodei este:

```
public int compareTo(String <altSir>)
```

unde:

- **<altSir>** - un alt obiect de tip **String** cu care se face comparația șirului curent.

2. Concatenarea șirurilor de caractere

Metoda compareTo returnează:

- *o valoare întreaga mai mica decat 0* dacă șirul curent este mai mic, din punct de vedere al ordinii lexicografice, decât șirul de caractere din parametrul `<altSir>`;
- *valoarea 0* dacă șirul curent este egal, din punct de vedere al ordinii lexicografice, cu șirul de caractere din parametrul `<altSir>`;
- *o valoare întreaga mai mare decat 0* dacă șirul curent este mai mare, din punct de vedere al ordinii lexicografice, decât șirul de caractere din parametrul `<altSir>`.

De exemplu:

```
int rezultat = sir1.compareTo(sir2);
```

2. Concatenarea șirurilor de caractere

C. Metoda compareToIgnoreCase este folosită pentru a *compara două șiruri de caractere, din punct de vedere al ordinii lexicografice, fără a face distincție între literele mici și mari ale alfabetului.*

Antetul metodei este:

```
public int compareToIgnoreCase(String <altSir>)
```

unde:

- **<altSir>** - un alt obiect de tip **String** cu care se face comparația șirului curent.

2. Concatenarea șirurilor de caractere

Determinarea lungimii șirurilor de caractere

- Lungimea unui obiect de tip `String` (un șir vid are lungimea 0) poate fi obținută cu metoda `length()`, care *returnează numărul de caractere, de tip Unicode, din șir.*

2. Concatenarea șirurilor de caractere

Java String Length Example

```
1.  /*
2.   Java String Length Example
3.   This example shows how to get a length of a given String object.
4.  */
5.
6.  public class StringLengthExample {
7.
8.   public static void main(String[] args) {
9.     //declare the String object
10.    String str = "Hello World";
11.
12.    //length() method of String returns the length of a String.
13.    int length = str.length();
14.    System.out.println("Length of a String is : " + length);
15.  }
16. }
17.
18. /*
19. Output of a program would be:
20. Length of a String is : 11
21. */
```

Clasa String in Java

1. Siruri de caractere (String-uri)
2. Concatenarea sirurilor de caractere
3. Alte metode din clasa String
4. Conversii intre siruri de caractere (String) si alte tipuri de date primitive

3. Alte metode din clasa String

- Extragerea unui caracter dintr-un sir de caractere - Metoda `charAt`
- Extragerea unui subsir dintr-un sir de caractere - Metoda `substring`
- Convertirea unui sir de caractere la un tablou de caractere - Metoda `toCharArray`

3. Alte metode din clasa String

- Cautarea primei aparitii a unui sir intr-un alt sir - Metoda `startsWith`
- Cautarea primei aparitii a unui sir intr-un alt sir folosind o pozitie de inceput a cautarii - Metoda `startsWith`

3. Alte metode din clasa String

Java String startsWith Example

<http://www.java-examples.com/>

```
1.  /*
2.   String startsWith Example
3.   This example shows how to check if a particular string is starting with
4.   a specified word.
5.  */
6.
7.  public class StringStartsWithExample {
8.
9.   public static void main(String[] args) {
10.
11.    //declare the original String
12.    String strOrig = "Hello World";
13.
14.    /*
15.    check whether String starts with Hello or not.
16.    Use startsWith method of the String class to check the same.
17.    startsWith() method returns true if a string is starting with a given word
18.    otherwise it returns false
19.    */
20.    if(strOrig.startsWith("Hello")){
21.        System.out.println("String starts with Hello");
22.    }else{
23.        System.out.println("String does not start with Hello");
24.    }
25.
26.    }
27. }
28.
29. /*
30. Output of the program would be :
31. String starts with Hello
32. */
```

3. Alte metode din clasa String

- Determinarea pozitiei primei aparitii a unui sir intr-un alt sir - Metoda **indexOf**
- Determinarea pozitiei primei aparitii a unui sir intr-un alt sir folosind o pozitie de inceput a cautarii - Metoda **indexOf**
- Cautarea ultimei aparitii a unui sir intr-un alt sir - Metoda **endsWith**
- Determinarea pozitiei ultimei aparitii a unui sir intr-un alt sir - Metoda **lastIndexOf**

3. Alte metode din clasa String

- Inlocuirea aparitiilor unui caracter dintr-un sir cu un alt caracter - Metoda **replace**
- Inlocuirea primei aparitii a unei secvente de caractere (subsir), dintr-un sir, cu o alta secventa de caractere (subsir) - Metoda **replaceFirst**
- Inlocuirea tuturor aparitiilor unei secvente de caractere (subsir), dintr-un sir, cu o alta secventa de caractere (subsir) - Metoda **replaceAll**

```
1.  /*
2.  Java String replace example.
3.  This Java String Replace example describes how replace method of java String class
4.  can be used to replace character or substring can be replaced by new one.
5.  */
6.
7.  public class JavaStringReplaceExample{
8.
9.  public static void main(String args[]){
10.
11.     /*
12.     Java String class defines three methods to replace character or substring from
13.     the given Java String object.
14.     1) String replace(int oldChar, int newChar)
15.     This method replaces a specified character with new character and returns a
16.     new string object.
17.     2) String replaceFirst(String regularExpression, String newString)
18.     Replaces the first substring of this string that matches the given regular
19.     expression with the given new string.
20.     3) String replaceAll(String regex, String replacement)
21.     Replaces the each substring of this string that matches the
22.     given regular expression with the given new string.
23.     */
24.
25.     String str="Replace Region";
26.
27.     /*
28.     Replaces all occurrences of given character with new one and returns new
29.     String object.
30.     */
31.     System.out.println( str.replace( 'R','A' ) );
32.
33.     /*
34.     Replaces only first occurrences of given String with new one and
35.     returns new String object.
36.     */
37.     System.out.println( str.replaceFirst("Re", "Ra" ) );
38.
39.     /*
40.     Replaces all occurrences of given String with new one and returns
41.     new String object.
42.     */
43.     System.out.println( str.replaceAll("Re", "Ra" ) );
44.
```

```
51. OUTPUT of the above given Java String Replace Example would be :
52.
53. Aeplace Aegion
54. Raplace Region
55. Raplace Ragion
56.
57. */
```

3. Alte metode din clasa String

Java String Trim Example

- Eliminarea caracterelor albe de la inceputul si sfarsitul unui sir de caractere - Metoda **trim**

```
1.  /*
2.   Java String Trim Example.
3.   This Java String trim example shows how to remove leading and trailing space
4.   from string using trim method of Java String class.
5.  */
6.
7.  public class RemoveLeadingTrailingSpace {
8.
9.      public static void main(String[] args) {
10.
11.         String str = " String Trim Example ";
12.
13.         /*
14.          * To remove leading and trailing space from string use,
15.          * public String trim() method of Java String class.
16.          */
17.
18.         String strTrimmed = str.trim();
19.
20.         System.out.println("Original String is: " + str);
21.         System.out.println("Removed Leading and trailing space");
22.         System.out.println("New String is: " + strTrimmed);
23.     }
24. }
25.
26. /*
27.  Output would be
28.  Original String is:  String Trim Example
29.  Removed Leading and trailing space
30.  New String is: String Trim Example
31.  */
```

3. Alte metode din clasa String

- Convertirea unui caracter la un sir de caractere - Metoda `valueOf`
- Convertirea unui tablou de caractere la un sir de caractere - Metoda `valueOf`
- Convertirea, in sirul de caractere, a literelor mari in litere mici - Metoda `toLowerCase`
- Convertirea, in sirul de caractere, a literelor mici in litere mari - Metoda `toUpperCase`

Clasa String in Java

1. Siruri de caractere (String-uri)
2. Concatenarea sirurilor de caractere
3. Alte metode din clasa String
4. Conversii intre siruri de caractere (String) si alte tipuri de date primitive

4. Conversia de la tipurile primitive de date numerice la tipul String și invers

- Tipul de data *byte* poate fi reprezentat in Java ca un obiect din **clasa Byte** (din pachetul *java.lang*).
 - Un obiect de tip **Byte** contine un singur camp al carui tip este *byte*.
- A. Pentru **conversia de la tipul de data *byte* la tipul de data *String*** se poate folosi **metoda `toString`** a clasei de obiecte `Byte`.

Antetul metodei este:

```
public static String toString(byte <b>)
```

- Metoda returneaza un nou sir de caractere continand valoarea precizata in parametrul .
- Baza de numeratie folosita este 10.

4. Conversia de la tipurile primitive de date numerice la tipul String și invers

B. Pentru **conversia de la tipul de data *String* la tipul de data *byte*** se poate folosi **metoda `parseByte`** a clasei de obiecte `Byte`.

Antetul metodei este:

```
public static byte parseByte(String <str>)
```

- Metoda returneaza reprezentarea de tipul ***byte***, in baza 10, a continutului sirului de caractere din parametrul <str>.
- Aceasta metoda genereaza o exceptie (eroare) daca *String-ul* de convertit nu contine o valoare de tip ***byte***.

4. Conversia de la tipurile primitive de date numerice la tipul String și invers

- **Metoda `parseByte`** poate fi folosită și pentru conversia unui șir de caractere într-o valoare de tip *byte* într-o baza de numeratie specificată.

Antetul metodei, în această formă, este:

```
public static byte parseByte(String <șir>, int <baza>)
```

Exemple:

```
String s1 = Byte.toString(45); // s1 conține valoarea "45"
```

```
byte x = Byte.parseByte("75",8); // x conține valoarea 75  
in baza 8
```

4. Conversia de la tipurile primitive de date numerice la tipul String și invers

- Tipul de data *Short* poate fi reprezentat în Java ca un obiect din **clasa Short** (din pachetul *java.lang*).
- Un obiect de tip **Short** conține un singur câmp al cărui tip este *short*.
- A. Pentru conversia de la tipul de data *short* la tipul de data *String* se poate folosi **metoda toString** a clasei de obiecte Short.

Antetul metodei este:

```
public static String toString(short <s>)
```

- Metoda returnează un nou șir de caractere conținând valoarea precizată în parametrul <s>.
- Baza de numeratie folosită este 10.

4. Conversia de la tipurile primitive de date numerice la tipul String și invers

B. Pentru conversia de la tipul de data *String* la tipul de data *short* se poate folosi **metoda `parseShort`** a clasei de obiecte `Short`.

Antetul metodei este:

```
public static short parseShort(String <str>)
```

- Metoda returneaza reprezentarea de tipul *short*, in baza 10, a continutului sirului de caractere din parametrul <str>.
- Aceasta metoda genereaza o exceptie (eroare) daca *String-ul* de convertit nu contine o valoare de tip *short*.

4. Conversia de la tipurile primitive de date numerice la tipul String și invers

- **Metoda `parseShort`** poate fi folosită și pentru conversia unui șir de caractere într-o valoare de tip *short*, într-o bază de numeratie specificată.

Antetul metodei, în această formă, este:

```
public static short parseShort(String <sir>,  
int <baza>)
```

Exemple:

```
String s1 = Short.toString(100); // s1 conține valoarea "100"  
short x = Short.parseShort("75", 2); // x conține valoarea 75  
in baza 2
```

4. Conversia de la tipurile primitive de date numerice la tipul String și invers

- Tipul de data *int* poate fi reprezentat in Java ca un obiect din **clasa Integer** (din pachetul *java.lang*).
 - Un obiect de tip **Integer** contine un singur camp al carui tip este *int*.
- A. Pentru conversia de la tipul de data *int* la tipul de data *String* se poate folosi **metoda toString** a clasei de obiecte Integer.

Antetul metodei este:

```
public static String toString(int <i>)
```

- Metoda returneaza un nou sir de caractere continand valoarea precizata de parametrul <i>.
- Baza de numeratie folosita este 10.

4. Conversia de la tipurile primitive de date numerice la tipul String și invers

- **Metoda toString** pentru numere de tip *int* poate fi folosita si sub forma:

public static String toString(int <i>, int <baza>)

- In acest caz, metoda returneaza un nou sir de caractere continand valoarea precizata de parametrul <i>, in baza de numeratie specificata in parametrul <baza>.

De exemplu:

```
String s1 = Integer.toString(20,2);
```

//s1 contine reprezentarea in baza 2 a numarului 20 sub forma de sir de caractere

4. Conversia de la tipurile primitive de date numerice la tipul String și invers

B. Pentru conversia de la tipul de data *String* la tipul de data *int* se poate folosi **metoda parseInt** a clasei de obiecte Integer.

Antetul metodei este:

```
public static int parseInt(String <str>)
```

- Metoda returneaza reprezentarea de tipul *int*, in baza 10, a continutului sirului de caractere din parametrul <str>.
- Aceasta metoda genereaza o exceptie (eroare) daca *String-ul* de convertit nu contine o valoare de tip *int*.

4. Conversia de la tipurile primitive de date numerice la tipul `String` și invers

- **Metoda `parseInt`** poate fi folosită și pentru conversia unui șir de caractere într-o valoare de tip *int*, într-o bază de numeratie specificată.

Antetul metodei, în această formă, este:

```
public static int parseInt(String <șir>, int <baza>)
```

- Următorul program ([ParseIntSiruri.java](#)) arată cum se face conversia unui șir de caractere (*String*) într-un număr de tip întreg (*int*) prin folosirea metodei **`Integer.parseInt`**.

```
import java.io.*;
public class ParseIntSiruri
{
public static void main(String[] args) throws IOException
{
    char[] caractere = new char[20];
    for (int i=0; i <=19; i++)
        caractere[i] = ' ';
    BufferedReader br1 = new BufferedReader(new
    InputStreamReader(System.in));
    System.out.println ("Introduceti primul numar");
    br1.read(caractere, 0, 20);
    String sir1= new String(caractere).trim();
    int x = Integer.parseInt(sir1);
    System.out.println("Sirul transformat in numar este:" + x);
    for (int i=0; i <=19; i++)
        caractere[i] = ' ';
```

```
System.out.println ("Introduceti al doilea  
numar");  
BufferedReader br2 = new BufferedReader(new  
InputStreamReader(System.in));  
br2.read(caractere, 0, 20);  
String sir2 = new String(caractere).trim();  
int y = Integer.parseInt(sir2);  
System.out.println("Sirul transformat in numar  
este:" + y);  
int z = x+y;  
System.out.println("Suma numerelor este:" + z);  
}  
}
```

```
curs10_2.java
1 package poo;
2 import java.io.*;
3
4 public class curs10_2 {
5
6     public static void main(String[] args) throws IOException {
7         char[] caractere = new char[20];
8         for (int i=0; i <=19; i++)
9             caractere[i] = ' ';
10        BufferedReader br1 = new BufferedReader(new InputStreamReader(System.in));
11        System.out.println ("Introduceti primul numar");
12        br1.read(caractere, 0, 20);
13        String sir1= new String(caractere).trim();
14        int x = Integer.parseInt(sir1);
15        System.out.println("Sirul transformat in numar este:" + x);
16        for (int i=0; i <=19; i++)
17            caractere[i] = ' ';
18        System.out.println ("Introduceti al doilea numar");
19        BufferedReader br2 = new BufferedReader(new InputStreamReader(System.in));
20        br2.read(caractere, 0, 20);
21        String sir2 = new String(caractere).trim();
22        int y = Integer.parseInt(sir2);
23        System.out.println("Sirul transformat in numar este:" + y);
24        int z = x+y;
25        System.out.println("Suma numerelor este:" + z);
26    }
27 }
```

Dupa executia programului pe ecran se afiseaza urmatoarele rezultate:

```
Problems @ Javadoc Declaration Console
<terminated> curs10_2 [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (11 nov. 2014, 15:25:26)
Introduceti primul numar
123
Sirul transformat in numar este:123
Introduceti al doilea numar
45678
Sirul transformat in numar este:45678
Suma numerelor este:45801
```

4. Conversia de la tipurile primitive de date numerice la tipul String și invers

- Tipul de data *long* poate fi reprezentat in Java ca un obiect din **clasa Long** (din pachetul *java.lang*).
- Un obiect de tip **Long** contine un singur camp al carui tip este *long*.
- A. Pentru conversia de la tipul de data *long* la tipul de data *String* se poate folosi **metoda toString** a clasei de obiecte **Long**, in cele doua variante descrise la tipul **Integer**.
- Modul de folosire a acestei metode este la fel ca la tipul de obiecte **Integer**.

4. Conversia de la tipurile primitive de date numerice la tipul String și invers

B. Pentru conversia de la tipul de data *String* la tipul de data *long* se poate folosi **metoda parseLong** a clasei de obiecte *Long*, in cele doua variante descrise la tipul *Integer*.

Modul de folosire a acestei metode este la fel ca la tipul de obiecte *Integer*.

De exemplu:

```
parseLong("473", 10) // returneaza 473L
```

```
parseLong("1100110", 2) // returneaza 102L
```

4. Conversia de la tipurile primitive de date numerice la tipul String și invers

- Tipul de data *float* poate fi reprezentat în Java ca un obiect din **clasa Float** (din pachetul *java.lang*).
- Un obiect de tip **Float** conține un singur câmp al cărui tip este *float*.

A. Pentru conversia de la tipul de data *float* la tipul de data *String* se poate folosi **metoda toString** a clasei de obiecte **Float**.

Antetul metodei este:

```
public static String toString(float <f>)
```

- Metoda returnează un nou șir de caractere conținând valoarea precizată de parametrul <f>.

De exemplu:

```
String s1 = Float.toString(20.3); // s1 conține valoarea "20.3"
```

4. Conversia de la tipurile primitive de date numerice la tipul String și invers

B. Pentru conversia de la tipul de data *String* la tipul de data *float* se poate folosi **metoda `parseFloat`** a clasei de obiecte `Float`.

Antetul metodei este:

```
public static float parseFloat(String <str>)
```

- Metoda returneaza reprezentarea de tipul *float*, a continutului sirului de caractere din parametrul <str>.
- Aceasta metoda genereaza o exceptie (eroare) daca *String-ul* de convertit nu contine o valoare de tip *float*.

De exemplu:

```
float y = Float.parseFloat("1.10"); // y contine valoarea numerica 1.10
```

4. Conversia de la tipurile primitive de date numerice la tipul String și invers

- Tipul de data *double* poate fi reprezentat in Java ca un obiect din **clasa Double** (din pachetul *java.lang*).
 - Un obiect de tip **Double** contine un singur camp al carui tip este *double*.
- A. Pentru conversia de la tipul de data *double* la tipul de data *String* se poate folosi **metoda toString** a clasei de obiecte **Double**.
- B. Pentru conversia de la tipul de data *String* la tipul de data *double* se poate folosi **metoda parseDouble** a clasei de obiecte **Double**.
- Modul de folosire a acestei metode este la fel ca la tipul de obiecte **Float**.

4. Conversia de la tipurile primitive de date numerice la tipul String și invers

- Tipul de data *char* poate fi reprezentat in Java ca un obiect din **clasa Character** (din pachetul *java.lang*).
- Un obiect de tip **Character** contine un singur camp al carui tip este *char*.
- Din clasa **Character** vom prezenta **metoda isDigit**, utilizata pentru a testa daca o valoare de tip *char* reprezinta o cifra sau nu.

4. Conversia de la tipurile primitive de date numerice la tipul String și invers

Metoda `isDigit` are urmatorul antet:

```
public static boolean isDigit(char <ch>)
```

- Metoda returneaza valoarea *true* daca parametrul <ch> este o cifra, altfel returneaza valoarea *false*.
- Aceasta metoda este important de folosit pentru conversiile de la tipul *String* la tipurile primitive de date, deoarece in cazul valorilor nenumeric, metodele de conversie prezentate “arunca” exceptii (dau erori).

Programul prezentat anterior se poate modifica astfel incat sa putem verifica daca datele introduse de la tastatura sunt numere sau nu (isDigitTest1.java).

```
import java.io.*;  
public class isDigitTest1  
{  
    public static void main(String[] args) throws  
IOException  
{  
    char[ ] caractere = new char[20];  
    for (int i=0; i <=19; i++)  
        caractere[i] = ' ';  
}  
}
```

```
BufferedReader br1 = new BufferedReader(new
InputStreamReader(System.in));
System.out.println ("Introduceti un numar");
br1.read(caractere, 0, 20);
int j = 0;
while (caractere [j] != '\r' )
{
    if ( !Character.isDigit(caractere[j]) )
    {
        System.out.println("Nu este numar");
        return;
    }
    j++;
}
}
```

```
curs10_3.java
1 package poo;
2 import java.io.*;
3
4 public class curs10_3 {
5
6     public static void main(String[] args) throws IOException {
7         char[] caractere = new char[20];
8         for (int i=0; i <=19; i++)
9             caractere[i] = ' ';
10        BufferedReader br1 = new BufferedReader(new InputStreamReader(System.in));
11        System.out.println ("Introduceti un numar");
12        br1.read(caractere, 0, 20);
13        int j = 0;
14        while (caractere [j] != '\r' )
15        {
16            if (!Character.isDigit(caractere[j]))
17            {
18                System.out.println("Nu este numar");
19                return;
20            }
21            j++;
22        }
23    }
24 }
```

Dupa executia programului pe ecran se afiseaza urmatoarele rezultate:

```
Problems @ Javadoc Declaration Console
<terminated> curs10_3 [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (11 nov. 2014, 15:36:44)
Introduceti un numar
45d3
Nu este numar
```

Program care verifica daca un sir de caractere este sau nu palindrom

```
package poo;
```

```
public class cus10_4 {  
    public static void main(String[] args) {  
        String palindrome = "Ele fac cafele";  
        int len = palindrome.length();  
        char[] tempCharArray = new char[len];  
        char[] charArray = new char[len];
```

```
        // sirul initial se pune intr-un  
        vector de caractere  
        for (int i = 0; i < len; i++) {  
            tempCharArray[i] =  
palindrome.charAt(i);  
        }
```

```
        // se inverseaza vectorul  
        for (int j = 0; j < len; j++) {  
            charArray[j] =  
tempCharArray[len - 1 - j];  
        }  
  
        String reversePalindrome =  
new String(charArray);  
  
        System.out.println(reversePalindr  
ome);  
    }  
}
```

Program care verifica daca un sir de caractere este sau nu palindrom

```
cus10_4.java ✕
1 package poo;
2
3 public class cus10_4 {
4
5     public static void main(String[] args) {
6         String palindrome = "Ele fac cafele";
7         int len = palindrome.length();
8         char[] tempCharArray = new char[len];
9         char[] charArray = new char[len];
10
11         // sirul initial se pune intr-un vector de caractere
12         for (int i = 0; i < len; i++) {
13             tempCharArray[i] = palindrome.charAt(i);
14         }
15
16         // se inverseaza vectorul
17         for (int j = 0; j < len; j++) {
18             charArray[j] = tempCharArray[len - 1 - j];
19         }
20
21         String reversePalindrome = new String(charArray);
22         System.out.println(reversePalindrome);
23     }
24 }
```

Dupa executia programului pe ecran se afiseaza urmatoarele rezultate:

```
Problems @ Javadoc Declaration Console ✕
<terminated> cus10_4 [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (11 nov. 2014, 16:19:19)
elefac caf eLE
```

Întrebări?