

Câteva clase și metode de bibliotecă

Dr. Petru Florin Mihancea

V20180924

1

Câteva metode mai speciale

Dr. Petru Florin Mihancea

Orice obiect are metodele ...

public boolean equals(Object o)

public String toString()

protected void finalize()

public int hashCode()

Sunt definite în clasa
Object și au o
implementare default

Dr. Petru Florin Mihancea

Cum verificăm **egalitatea** ?

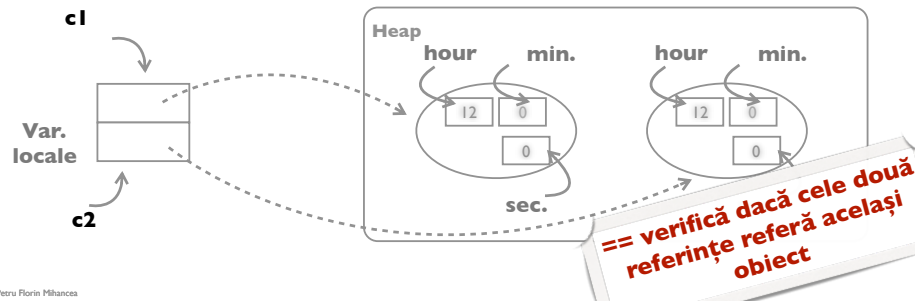
```
class Clock {  
    private int hour, minute, seconds;  
  
    public Clock(int h, int m, int s) {  
        setTime(h, m, s);  
    }  
  
    public void setTime(int h, int m, int s) {  
        hour = (h >= 0) && (h < 24) ? h : 0;  
        minute = (m >= 0) && (m < 60) ? m : 0;  
        seconds = (s >= 0) && (s < 60) ? s : 0;  
    }  
  
    public void print() {  
        System.out.println("Current time " + hour + ":" + minute + ":" + seconds);  
    }  
}
```

Dr. Petru Florin Mihancea

Exemplu

```
class Main {  
    public static void main(String[] args) {  
        Clock c1 = new Clock(12, 0, 0);  
        Clock c2 = new Clock(12, 0, 0);  
        System.out.println(c1 == c2);  
    }  
}
```

Output
false



Dr. Petru Florin Măhănea

Două feluri de “egalitate”

Fizică sau de identitate
folosim == (!= la inegalitate)

Egale din punctul de vedere al stării
definim o metodă pentru acest lucru

și nu orice metodă !!! În
Java această egalitate se
va implementa în metoda
equals

Dr. Petru Florin Măhănea

Cum scriem equals ?

```
class Clock {  
    private int hour, minute, seconds;  
    ...  
    public boolean equals(Object o) {  
        if(o instanceof Clock) {  
            Clock comparingClock = (Clock)o;  
            return hour == comparingClock.hour &&  
                minute == comparingClock.minute &&  
                seconds == comparingClock.seconds;  
        }  
        return false;  
    }  
}
```

Implementarea default a
lui equals (în Object)
echivalent ==

```
class Main {  
    public static void main(String[] args) {  
        Clock c1 = new Clock(12, 0, 0);  
        Clock c2 = new Clock(12, 0, 0);  
        System.out.println(c1 == c2);  
        System.out.println(c1.equals(c2));  
    }  
}
```

Output
false
true

Dr. Petru Florin Măhănea

equals trebuie să fie ...

reflexiv

x.equals(x) e true

simetric

x.equals(y) e true și y.equals(x) e true

tranzitiv

x.equals(y) e true și y.equals(z) e true
atunci x.equals(z) e true

x.equals(y) întoarce aceeași valoare la apeluri
repetate (fără schimbarea stării obiectelor)

x.equals(null) e false

Dr. Petru Florin Măhănea

public int hashCode()

Întoarce codul de hashing al obiectului
vedem mai târziu

Țineți minte

$x.equals(y) \rightarrow x.hashCode() == y.hashCode()$

Când schimbăm equals
trebuie schimbat și
hashCode pt. a putea
lucra cu biblioteca ce se
bazează pe hashing

Dr. Petru Florin Măhăceș

public String toString()

**Întoarce reprezentarea sub formă de
șir de caractere pt. obiectul respectiv**

```
class Clock {  
    ....  
    public String toString() {  
        return "Current time " + hour + ":" + minute + ":" + seconds;  
    }  
    public void print() {  
        System.out.println("Current time " + hour + ":" + minute + ":" + seconds);  
    }  
}
```

Dr. Petru Florin Măhăceș

Mulți se bazează pe toString

```
PrintStream  
...  
+print(b : boolean) : void  
+print(c : char) : void  
+print(i : int) : void  
+print(s : double) : void  
+print(s : String) : void  
...  
+println(b : boolean) : void  
+println(c : char) : void  
+println(i : int) : void  
+println(s : double) : void  
+println(s : String) : void  
+println(o : Object) : void  
...
```

(+) string concatenation

```
class Main {  
    public static void main(String[] args) {  
        Clock c1 = new Clock(12, 0, 0);  
        System.out.println(c1);  
        String tmp = "The clock referred by c1 is " + c1;  
        System.out.println(tmp);  
    }  
}
```

Output
Current time 12:0:0
The clock referred by c1 is
Current time 12:0:0

Dr. Petru Florin Măhăceș

protected void finalize()

**Apelată (o singură dată) de colectorul de deșeuri
când acesta determină că obiectul nu mai poate fi
referit din program**

```
class Clock {  
    ....  
    protected void finalize() {  
        System.out.println("Gata :(");  
    }  
}
```

```
class GC {  
    public static void main(String[] args) {  
        for(int i = 0; i < 10000000; i++) {  
            new Clock(12,0,0);  
        }  
    }  
}
```

Pentru a realiza eliberare
de resurse

Dr. Petru Florin Măhăceș

2

String

Nu e tip primitiv!
De fapt șirurile de
caractere în Java sunt
instanțe ale clasei **String**

Dr. Petru Florin Măhăneanu

String-uri (I)

```
String
+String()
+String(original:String)
...
+charAt(index : int) : char
+concat(str : String) : String
+endsWith(suffix : String) : boolean
+equals(o : Object) : boolean
+indexOf(str : String) : int
+indexOf(ch : int) : int
+intern() : String
+lastIndexOf(ch : int) : int
+lastIndexOf(str : String) : int
+length() : int
+startsWith(prefix : String) : boolean
+substring(beginIndex : int) : String
+toUpperCase() : String
...
+valueOf(b : boolean) : String
+valueOf(i : int) : String
+valueOf(d : double) : String
...
```

```
String s1 = "Test";
String s2 = new String("Test");
System.out.println("Test".length());
```

 **Literalii șiruri din program sunt referințe la instanțe String, iar variabilele String sunt referințe la obiecte**

Dr. Petru Florin Măhăneanu

String-uri (II)

```
String
+String()
+String(original:String)
...
+charAt(index : int) : char
+concat(str : String) : String
+endsWith(suffix : String) : boolean
+equals(o : Object) : boolean
+indexOf(str : String) : int
+indexOf(ch : int) : int
+intern() : String
+lastIndexOf(ch : int) : int
+lastIndexOf(str : String) : int
+length() : int
+startsWith(prefix : String) : boolean
+substring(beginIndex : int) : String
+toUpperCase() : String
...
+valueOf(b : boolean) : String
+valueOf(i : int) : String
+valueOf(d : double) : String
...
```

```
String a1 = "Test ";
String a2 = new String("stringuri");
String a3 = a1.concat(a2);
System.out.println(a1);
System.out.println(a2);
System.out.println(a3);
```

Output
Test
stringuri
Test stringuri

 **Obiectele string sunt imutabile**
(Un obiect ce nu-și schimbă starea odată creat se spune că e imutabil)

Dr. Petru Florin Măhăneanu

Egalitatea

```
class Main {
    public static void main(String argv[]) {
        String e1 = "Sir 1";
        String e2 = "Sir 1";
        String e3 = new String("Sir 1");
        String e4 = new String("Altceva");

        System.out.println(e1 == e3);
        System.out.println(e1.equals(e3));
        System.out.println(e1 == e2);
        System.out.println(e3 == e4);
        System.out.println(e1.equals(e4));
    }
}
```

OUTPUT
false
true
true
false
false

Dr. Petru Florin Măhăneanu

3

Clase înfășurătoare

Orice poate fi obiect

Dr. Petru Florin Mihai

Necesare pt. că (de exemplu) anumite facilități de bibliotecă lucrează numai cu obiecte

Tip primitiv

Clasa înfășurătoare

byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

int-ul primitiv
5

5

Obiect al clasei
Integer

Aceste obiecte sunt
imutabile

Dr. Petru Florin Mihai

Exemplu Integer

Integer
...
+Integer(value : int)
+Integer(s : String)
+intValue() : int
+compareTo(i : Integer) : int
+equals(o : Object) : boolean
+parseInt(s : String) : int
+valueOf(i : int) : Integer
...

```
class Integers {
    public static void main(String[] args) {
        Integer i1 = new Integer(5);
        Integer i2 = new Integer(5);
        System.out.println(5 == 5);
        System.out.println(i1 == i2);
        System.out.println(i1.equals(i2));
    }
}
```

Output

true
false
true

Dr. Petru Florin Mihai

Tipuri primitive ca și obiecte

Unele “probleme”

```
class Exemplu {
    public static void addFive(Integer x) {
        int a = 5;
        int b = a + x;
        System.out.println(b);
    }
    public static void main(String[] args) {
        addFive(5);
    }
}
```

Output

10

Se da conversie de la Integer la int in mod automat

dar numai de la Java ≥ 1.5

Dr. Petru Florin Mihai

Java < 1.5

```
class Example {  
  
    public static void addFive(Integer x) {  
        int a = 5;  
        int b = a + x.intValue();  
        System.out.println(b);  
    }  
  
    public static void main(String[] args) {  
        addFive(Integer.valueOf(5));  
    }  
}
```

Dr. Petru Florin Mihances

Autoboxing & Unboxing

```
class Exemplu {  
    public static void addFive(Integer x) {  
        int a = 5;  
        int b = a + x;  
        System.out.println(b);  
    }  
    public static void main(String[] args) {  
        addFive(5);  
    }  
}
```

Unboxing: extragerea automată a unei valori primitive (ex. int) dintr-o instanță a clasei corespunzătoare (ex. Integer)

Autoboxing: înfășurarea automată a unei valori primitive (ex. int) într-o instanță a clasei corespunzătoare (ex. Integer)

Java ≥ 1.5

Operațiile altădată scrise explicit sunt introduse automat de compilator

Dr. Petru Florin Mihances

4

Tablouri

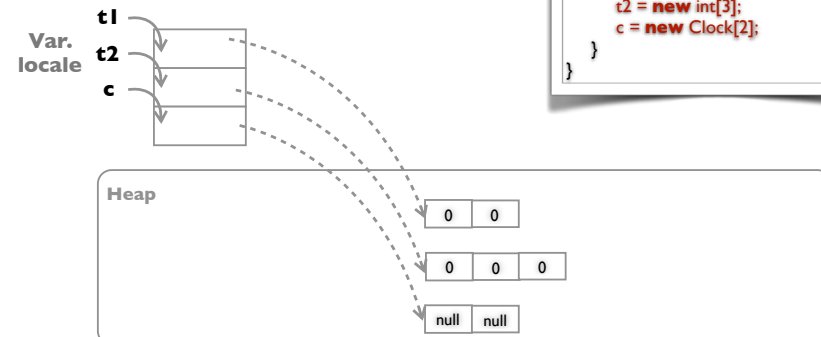
Don't shoot ... cum ziceam, orice poate fi obiect

Dr. Petru Florin Mihances

Crearea și referirea tablourilor

la rularea programului prin operatorul **new**, sunt alocate în **heap** și accesate prin **variabile referință**

```
class Main {  
    public static void main(String argv[]) {  
        //TipIntrare[] numeReferinta;  
        int[] t1, t2;  
        Clock[] c;  
        //new TipulIntrarii[dimensiune]  
        t1 = new int[2];  
        t2 = new int[3];  
        c = new Clock[2];  
    }  
}
```

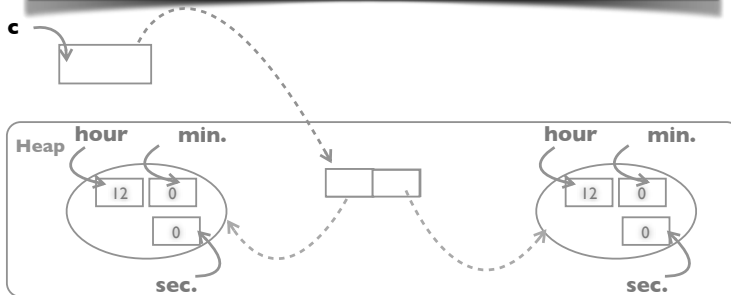


Dr. Petru Florin Mihances

Accesarea și parcurgerea

```
class Main {
    public static void main(String argv[]) {
        //câmpul length - nr. intrări alocate
        //și nu se mai poate schimba după alocare
        Clock[] c = new Clock[2];
        //accesul se face prin operatorul [ index ]
        //iar prima locație e la 0
        for(int i = 0; i < c.length; i++) {
            c[i] = new Clock(12,0,0);
        }
    }
}
```

Var.
locale



Dr. Petru Florin Mîhăneasa

Atenție la ...

```
class Tab1 {
    public static void main(String[] args) {
        Clock c[] = null;
        c[0] = new Clock(0,0,0);
    }
}
```

Output
Exception in thread "main"
java.lang.NullPointerException
at Tab1.main(Tab1.java:5)

```
class Tab2 {
    public static void main(String[] args) {
        Clock c[];
        c = new Clock[2];
        for(int i = 0; i < c.length + 1; i++) {
            c[i] = new Clock(12,0,0);
        }
    }
}
```

Output
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 2
at Tab2.main(Tab2.java:7)

Dr. Petru Florin Mîhăneasa

Atenție la...

```
class Tab3 {
    public static void main(String[] args) {
        int[] t;
        Clock c[];
        t = new int[2];
        c = new Clock[2];
        System.out.println("t[0]=" + t[0]);
        System.out.println("c[0]" + c[0].toString());
    }
}
```

Output
t[0]=0
Exception in thread "main"
java.lang.NullPointerException
at Tab3.main(Tab3.java:9)

Fiecare tablou are 0 pe pozitii la initializare, daca e numeric
Iar daca nu, are null pe fiecare pozitie

Dr. Petru Florin Mîhăneasa

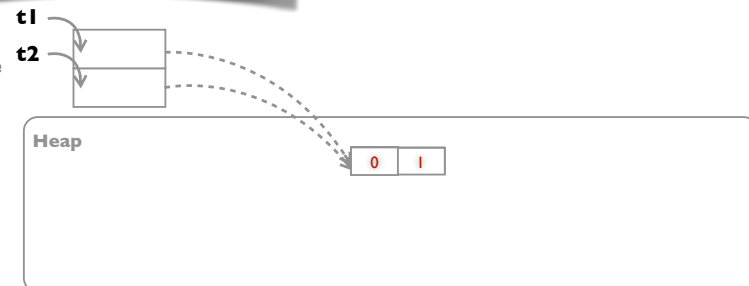
Atenție la...

```
class Tab3 {
    public static void main(String[] args) {
        int[] t1,t2;
        t1 = new int[2];
        for(int i = 0; i < t1.length; i++) {
            t1[i] = -i;
        }
        t2 = t1;
        for(int i = 0; i < t2.length; i++) {
            t2[i] = i;
        }
        for(int i = 0; i < t1.length; i++) {
            System.out.print(t1[i] + " ");
        }
    }
}
```

Output

0 1

Var.
locale



Dr. Petru Florin Mîhăneasa

Alt mod de inițializare

```
class Tab5 {  
    public static void main(String[] args) {  
        //Inițializatori de tablouri  
        // { exp0, exp1, ..., expN-1 }; (când se declară și referința)  
        // sau new TipIntrare[] { exp0, exp1, ..., expN-1 };  
        Clock[] c = {new Clock(0,0,0), new Clock(1,0,0), new Clock(2,0,0)};  
        for(int i = 0; i < c.length; i++) {  
            System.out.println(c[i]);  
        }  
    }  
}
```

Dr. Petru Florin Mîhăneș

... și parcurgere

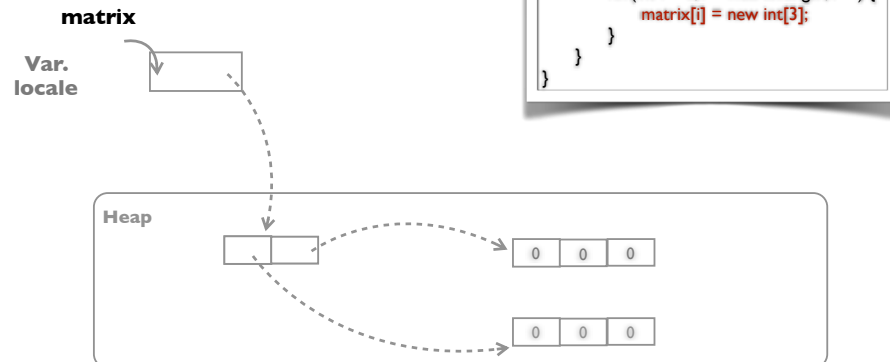
```
class Tab6 {  
    public static void main(String[] args) {  
        Clock[] c = new Clock[] {new Clock(0,0,0),new Clock(1,0,0),new Clock(2,0,0)};  
        //for(TipIntrare numeVar : RefTablou) { ... }  
        for(Clock aClock : c) {  
            System.out.println(aClock);  
        }  
    }  
}
```

Output
Current time 0:0:0
Current time 1:0:0
Current time 2:0:0

Dr. Petru Florin Mîhăneș

Tablouri multi-dimensionale

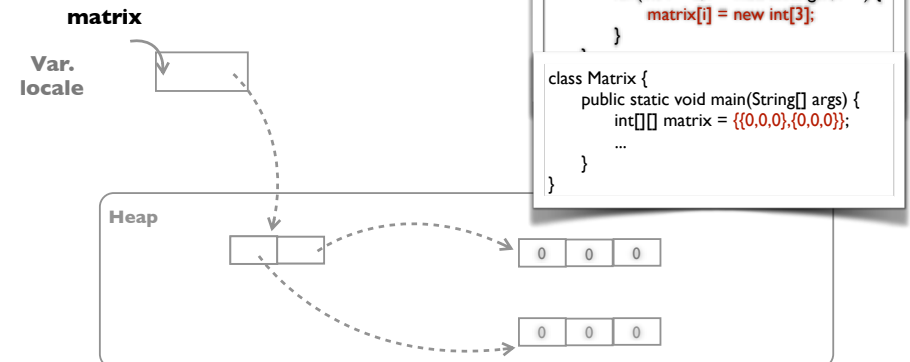
**o intrare dintr-un
tablou poate fi la rândul
ei referință la un tablou**



Dr. Petru Florin Mîhăneș

Tablouri multi-dimensionale

**o intrare dintr-un
tablou poate fi la rândul
ei referință la un tablou**



Dr. Petru Florin Mîhăneș

Parcurgeri **multi-dimensionale**

`int[][] matrix = ...`

```
for(int i = 0; i < matrix.length; i++) {  
    for(int j = 0; j < matrix[i].length; j++) {  
        System.out.print(matrix[i][j] + " ");  
    }  
    System.out.println();  
}
```

```
for(int[] aLine : matrix) {  
    for(int aCell : aLine) {  
        System.out.print(aCell + " ");  
    }  
    System.out.println();  
}
```

Dr. Petru Florin Mîhăneasa

Clasa **Arrays**

Arrays
...
<u>+toString(a : int[]) : String</u>
<u>+toString(a : Object[]) : String</u>
...
<u>+sort(a : int[]) : void</u>
...
<u>+equals(a : int[], a2 : int[]) : boolean</u>
<u>+equals(a : Object[], a2 : Object[]) : boolean</u>
...
<u>+copyOf(orig : int[], newLength : int) : int[]</u>
...

Tot felul de metode statice utilitar.
Atenție: nu e clasa obiectelor tablou

Dr. Petru Florin Mîhăneasa

Quiz

Câte obiecte **Integer** se creează la execuția liniilor de mai jos ?

```
Integer[] tab;  
tab = new Integer[10];
```

Când prelucrăm elementele unui tablou,
prelucrăm tot timpul toate intrările ?
(mergem tot timpul până la `tab.length` ?)

Dr. Petru Florin Mîhăneasa

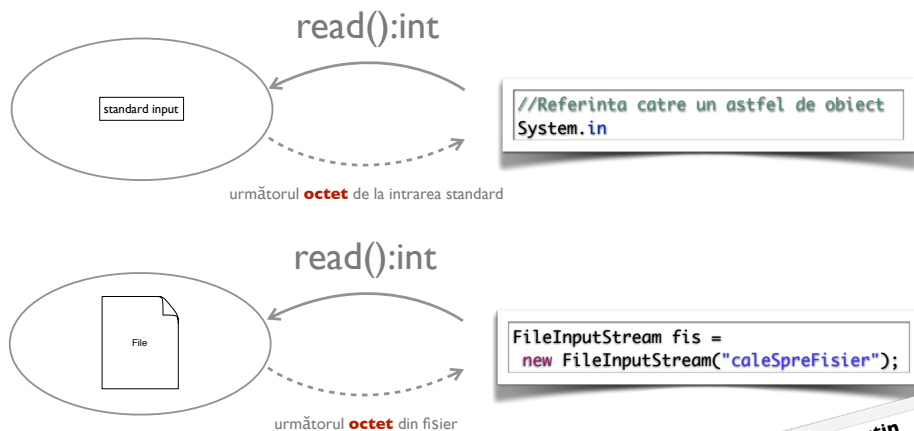
5

Elemente de **I/O**

Dr. Petru Florin Mîhăneasa

Flux de **intrare** orientat pe octet

InputStream



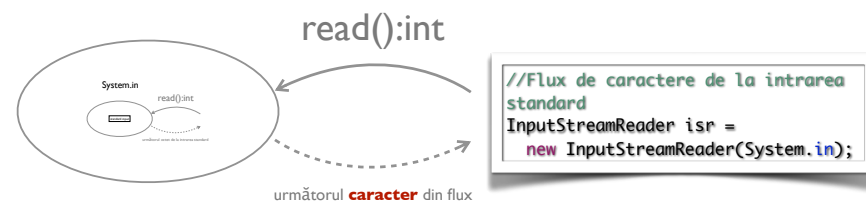
read întorce -1 la sfârșit de flux

Datele sunt în cel mai puțin semnificativ octet. Metoda close() închide fluxul.

Dr. Petru Florin Mihănea

Flux de **intrare** orientat pe caracter

InputStreamReader

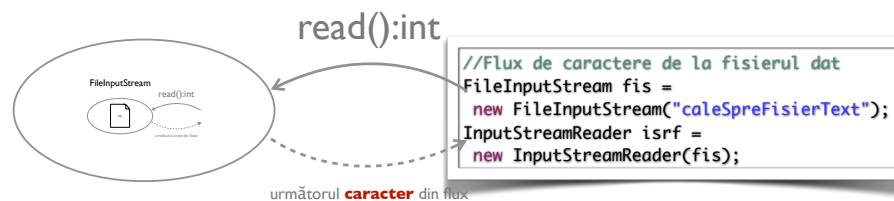


Dar de unde are
octeții care codifică
un caracter ?

Dr. Petru Florin Mihănea

Flux de **intrare** orientat pe caracter

InputStreamReader



Dar de unde are
octeții care codifică
un caracter ?

read întorce -1 la sfârșit de flux

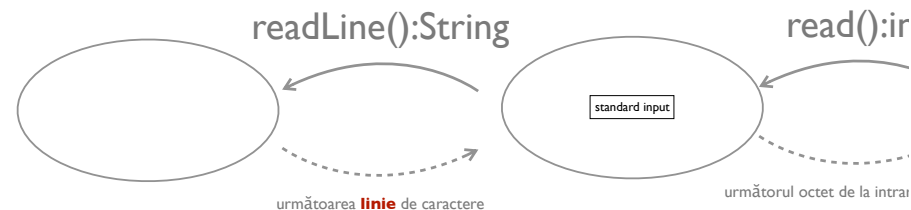
Datele sunt în cei mai puțin semnificativi 2 octeți. Metoda close() închide fluxul.

Dr. Petru Florin Mihănea

Flux de **intrare** orientat pe linie

BufferedReader

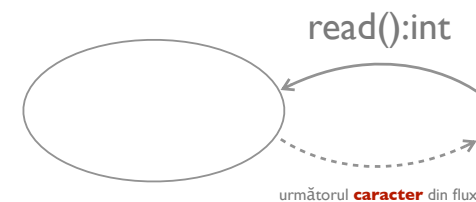
System.in



Dar de unde are
caracterele ?

Dar de unde are
octeții ?

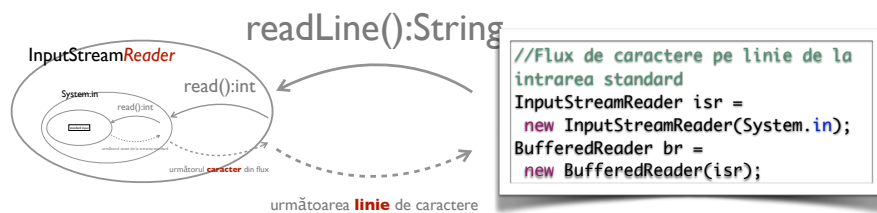
InputStreamReader



Dr. Petru Florin Mihănea

Flux de **intrare** orientat pe linie

BufferedReader



Dar de unde are caracterele ?

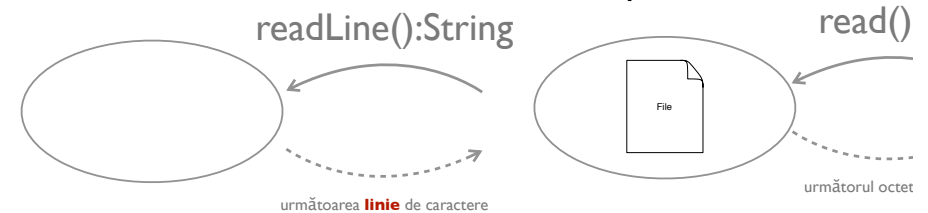
Dar de unde are octeții ?

Dr. Petru Florin Mihăneș

Flux de **intrare** orientat pe linie

BufferedReader

FileInputStream



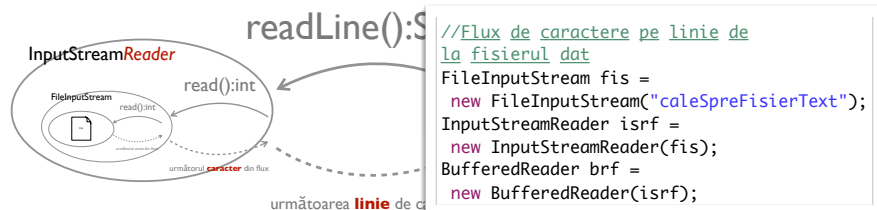
Dar de unde are caracterele ?

Dar de unde are octeții ?

Dr. Petru Florin Mihăneș

Flux de **intrare** orientat pe linie

BufferedReader



Dar de unde are caracterele ?

Dar de unde are octeții ?

Dr. Petru Florin Mihăneș

readLine întorce null la sfârșit de flux.
Metoda close() închide fluxul

Flux de **intrare** orientat pe repr. binare

```
//Flux de intrare orientat pe date binare
FileInputStream fis = new FileInputStream("caleSpreFisierBinar");
DataInputStream dis = new DataInputStream(fis);
```

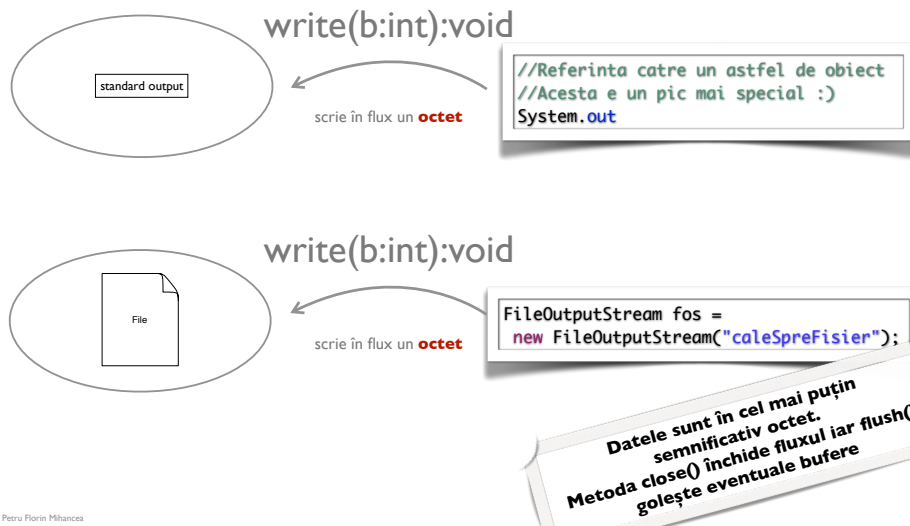
DataInputStream
+DataInputStream(in : InputStream)
...
+readByte() : byte
+readBoolean() : boolean
+readChar() : char
+readDouble() : double
+readFloat() : float
+readInt() : int
+readUTF() : String
...

Metoda close() închide fluxul

Dr. Petru Florin Mihăneș

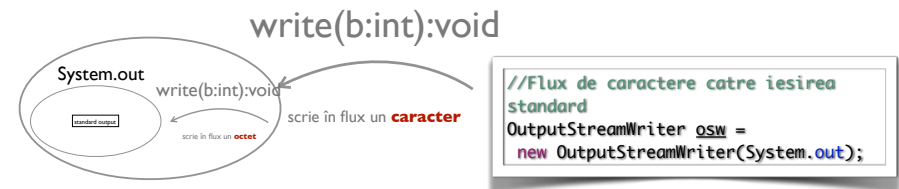
Flux de ieşire orientat pe octet

OutputStream



Flux de ieşire orientat pe caracter

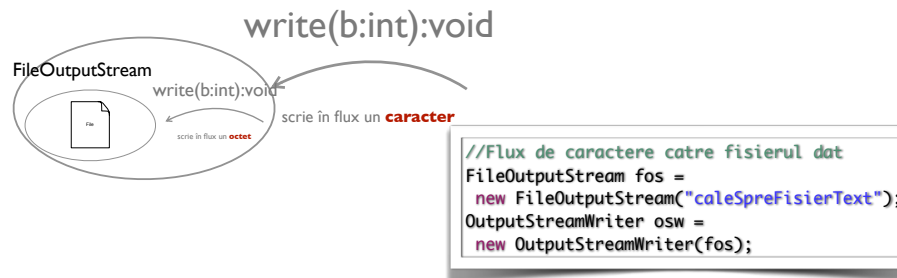
OutputStreamWriter



Dar unde scrie octeții caracterului ?

Flux de ieşire orientat pe caracter

OutputStreamWriter

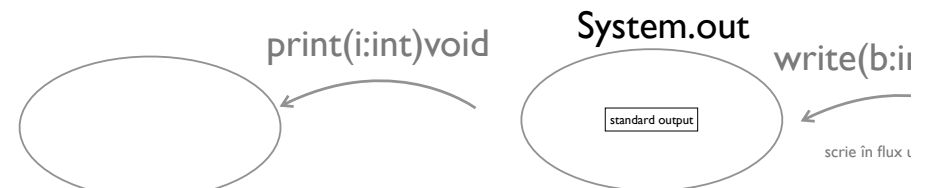


Dar unde scrie octeții caracterului ?

Datele sunt în cei mai puțin semnificativi 2 octeți. Metoda `close()` închide fluxul iar `flush()` golește eventuale bufer.

Flux de ieşire orientat pe linie

PrintWriter



Dar unde scrie caracterele ?

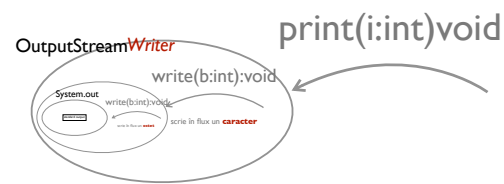
OutputStreamWriter



Dar unde scrie octeții ?

Flux de ieşire orientat pe linie

PrintWriter



```
//Flux de caractere pe linie la
//iesirea standard
OutputStream os0 = System.out;
OutputStreamWriter osw0 =
new OutputStreamWriter(os0);
PrintWriter ps0 =
new PrintWriter(osw0);
```

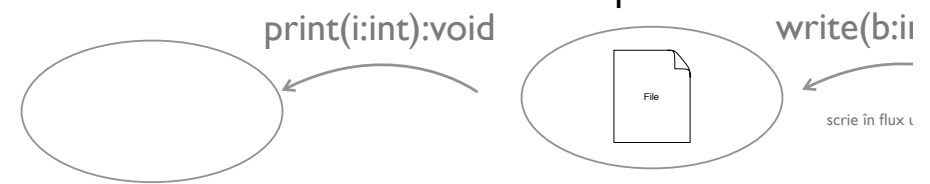
Dar unde scrie
caracterele ?

Dar unde scrie octeţii
?

Dr. Petru Florin Mihăneasa

Flux de ieşire orientat pe linie

PrintWriter



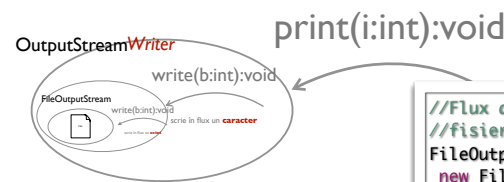
Dar unde scrie
caracterele ?

Dar unde scrie octeţii
?

Dr. Petru Florin Mihăneasa

Flux de ieşire orientat pe linie

PrintWriter



```
//Flux de caractere pe linie la
//fişierul dat
FileOutputStream os1 =
new FileOutputStream("caleSpreFişierText");
OutputStreamWriter osw1 =
new OutputStreamWriter(os1);
PrintWriter ps1 = new PrintWriter(osw1);
```

PrintWriter
+PrintWriter(out:Writer)
+close():void
+flush():void
+print(b:boolean):void
+print(str:String):void
+println():void
+print(i:int):void
...

Dar unde scrie
caracterele ?

Dar unde scrie octeţii
?

Dr. Petru Florin Mihăneasa

Flux de ieşire orientat pe repr. binare

```
//Flux de iesire orientat pe date binare
FileOutputStream fos = new FileOutputStream("caleSpreFişierBinar");
DataOutputStream dos = new DataOutputStream(fos);
```

DataOutputStream
+DataOutputStream(out : OutputStream)
...
+writeByte(b : int) : void
+writeBoolean(b : boolean) : void
+writeChar(v : int) : void
+writeDouble(v : double) : void
+writeFloat(v : float) : void
+writeInt(v : int) : void
+writeUTF(v : String) : void
...

Metoda close() închide fluxul.
Metoda flush() forţează golirea
eventualelor bufere

Dr. Petru Florin Mihăneasa

Exemplu

```
import java.io.*;

class IOExemplu {
    public static void main(String[] args) {
        try {
            BufferedReader bf = new BufferedReader(
                new InputStreamReader(
                    new FileInputStream("ExempluIN.txt")));
            DataOutputStream dos = new DataOutputStream(
                new FileOutputStream("ExempluOUT.dat"));
            String line;
            while((line = bf.readLine()) != null) {
                int readInt = Integer.parseInt(line);
                dos.writeInt(readInt);
            }
            bf.close();
            dos.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

import, try, catch -
vom învăța mai târziu

Flux de ieșire (caractere) mai altfel

PrintStream
+PrintStream(out:OutputStream)
+print(v:boolean):void
+println(v:boolean):void
+print(v:String):void
+println(v:String):void
+print(v:Object):void
+println(v:Object):void
...

System.out e o astfel de referință

- se comportă și ca un OutputStream dar seamănă și cu PrintWriter
- face flushing automat ex. la println, \n, etc.
- a rămas așa pt. că e de la prima versiune de Java

O grămadă de alte clase ce construiesc ele
structurile amintite în curs (și nu numai)
+ alte operații pentru simplificarea
activității
de programare ex. Scanner