

$obj_1 == obj_2$

↓
verifică dacă referințele sunt
egale

(duc la aceeași zonă de mem.)

2 feluri de egalitate

- fizică / identitate $==$
- din pct. de vedere a stării

→ definim o metodă
(equals)

public boolean equals(Object o)

```
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;  
    }  
}
```

instance
check ✓ void ✗

type cast

verificarea
proprie și a
celui alt

Implementarea default al
lui equals (in 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

HashCode

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

toString

```
class Clock {
```

```
....
```

```
public String toString() {
```

```
    return "Current time " + hour + ":" + minute + ":" + seconds;
```

```
}
```

```
public void print() {
```

```
    System.out.println("Current time " + hour + ":" + minute + ":" + seconds);
```

```
}
```

```
}
```

`System.out.println(clock);`
→ overloaded method

X

+ concatenare de String-uri

`String tmp = "ceva ceva" + clock;`

Finalize

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

String-uri

Imutabil → nu își poate schimba starea
(dacă clasă are set-er → mutabil)
Stringurile → imutabile

```
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

Wrapper Classes

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



Obiect al clasei
Integer

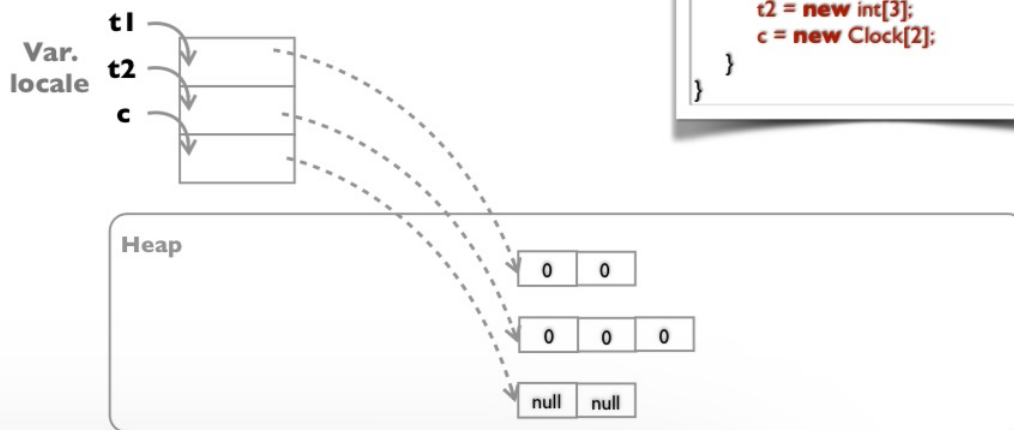
Aceste obiecte sunt
imutabile

Tablouri

Crearea și referirea tablourilor

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

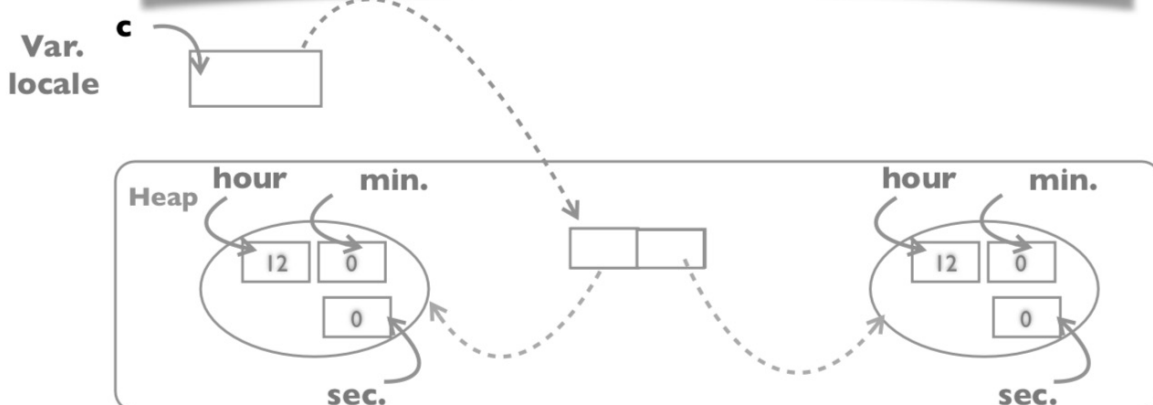
```
class Main {  
    public static void main(String argv[]) {  
        //TipIntrare[] numeReferinta;  
        int[] t1, t2;  
        Clock[] c;  
        //new TipIntrarii[dimensiune]  
        t1 = new int[2];  
        t2 = new int[3];  
        c = new Clock[2];  
    }  
}
```



pt. val. primitive \rightarrow init = 0
pt. obiecte \rightarrow init = null

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);  
        }  
    }  
}
```



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)

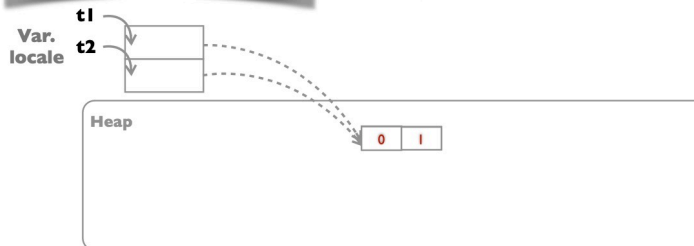
Dr. Petru Florin Mihaicea

```
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] + " ");  
        }  
    }  
}
```

Atenție la...

Output

0 1



Dr. Petru Florin Mihaicea

Elemente de I/O

- Flux de intrare orientat pe OCTET
→ InputStream (secvență de octeți)

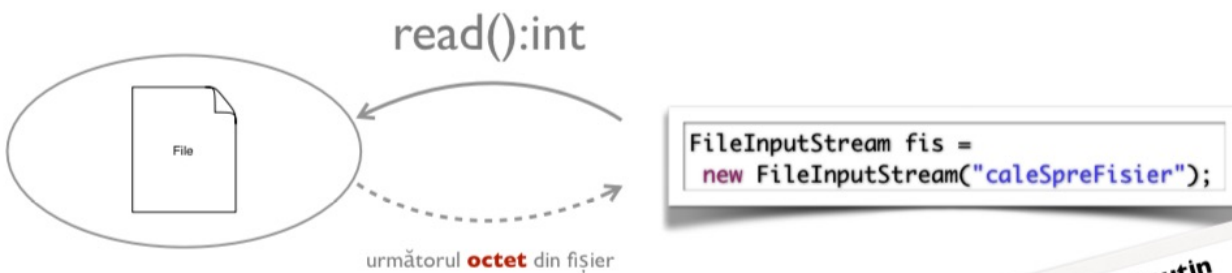
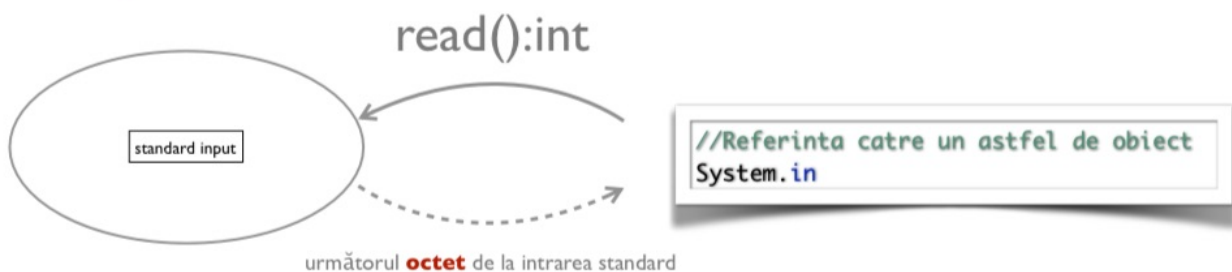
legat de Standard Input / File

→ read(): int → octet
— 1 → end

System.in

Flux de **intrare** orientat pe octet

InputStream

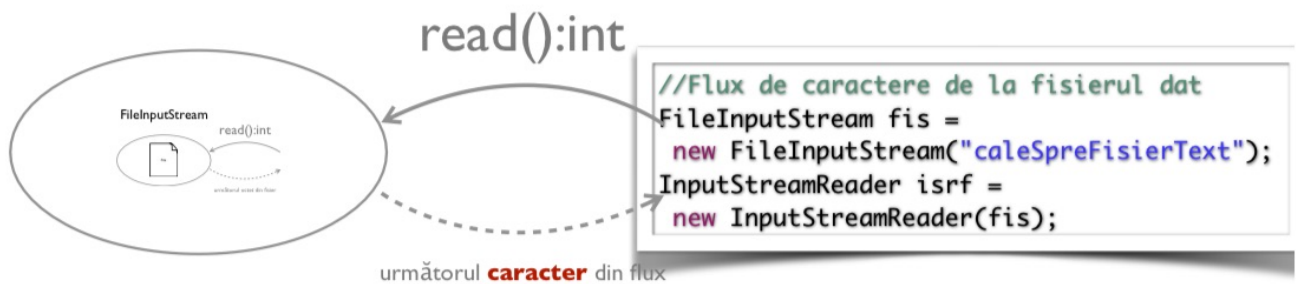


read întorce -1 la sfârșit de flux

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

Flux de **intrare** orientat pe caracter

InputStream**Reader**



Dar de unde are
octeții care codifică
un caracter ?

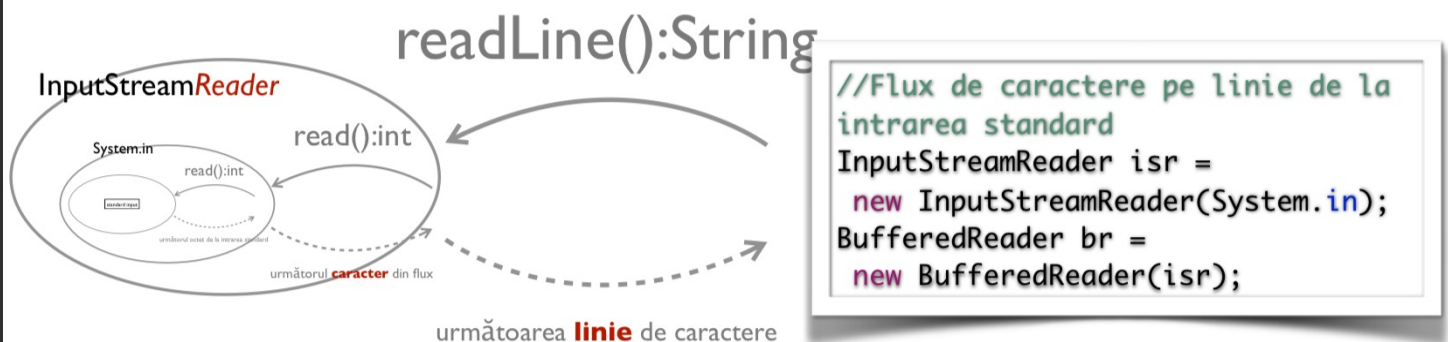
read întorce -1 la sfârșit de flux

Dr. Petru Florin Mihancea

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

Flux de **intrare** orientat pe linie

Buffered**Reader**



Flux de ieşire orientat pe octet

OutputStream



Flux de ieşire orientat pe caracter

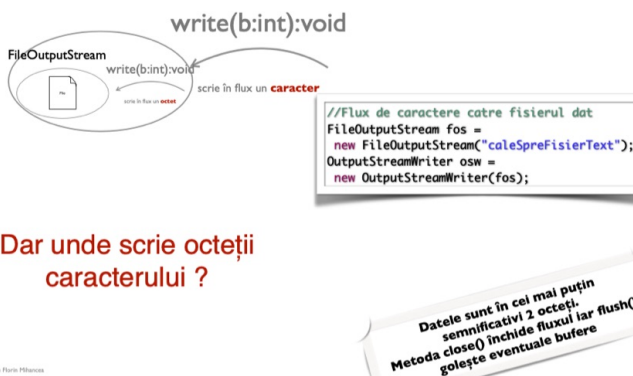
OutputStreamWriter



Dar unde scrie octeții
caracterului ?

Flux de ieşire orientat pe caracter

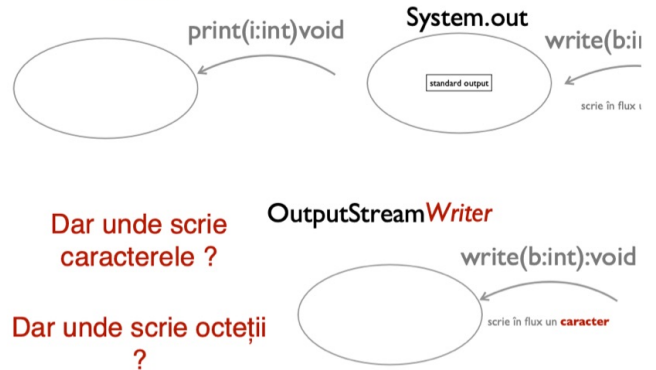
OutputStreamWriter



Dar unde scrie octeții
caracterului ?

Flux de ieşire orientat pe linie

PrintWriter

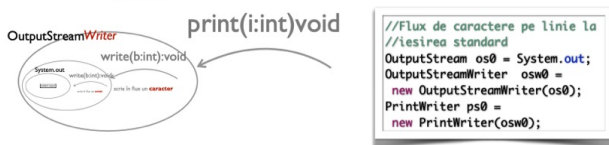


Dar unde scrie
caracterele ?

Dar unde scrie octeții
?

Flux de ieşire orientat pe linie

PrintWriter

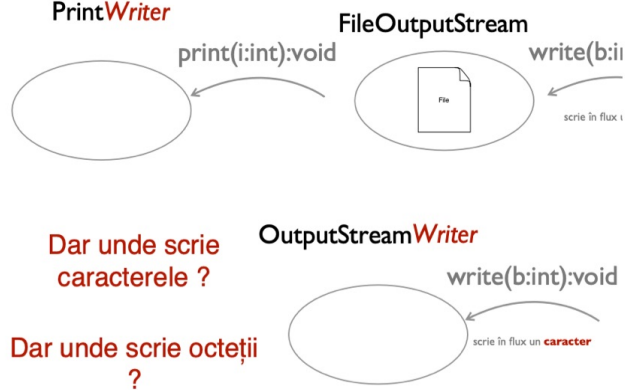


Dar unde scrie
caracterele ?

Dar unde scrie octeții
?

Flux de ieşire orientat pe linie

PrintWriter

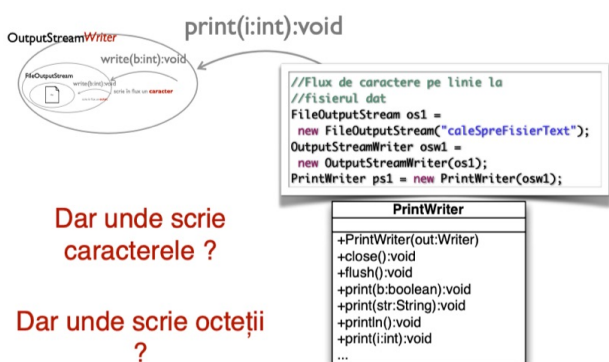


Dar unde scrie
caracterele ?

Dar unde scrie octeții
?

Flux de ieşire orientat pe linie

PrintWriter



Dar unde scrie
caracterele ?

Dar unde scrie octeții
?

Flux de ieşire orientat pe repr. binare

//Flux de ieșire orientat pe date binare
FileOutputStream fos = new FileOutputStream("caleSpreFisierBinar");
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() forțează golirea
eventualelor bufere*