Java

Actualizat 9.10.2018

Prima clasa

```
class Aplicatie1 {

    public static void main( String args[] )  {
        System.out.println( "Succes !" );
    }
}
```

Argumente main()

```
class Aplicatie2
{
 public static void main( String args[] )  {
    int i;
    System.out.println( "Numarul de argumente este "
                                        + args.length +  " :");
    for(i=0; i<args.length;  i++)
              System.out.println (i+  " -  "  +  args[i] );
  }
}
```

Tipuri de aplicatii Java

Toate aplicatiile Java sunt programe ce cuprind o lista de clase.

In urma compilarii rezulta un "cod de octeti" - bytecode - care poate fi executat

- pe o masina independenta (**aplicatii standalone**)
- de catre un browser dupa preluarea de pe retea (**applet-uri**)

Tipuri de date

Tipul primitiv	Dimensiune	Valoare minimă	Valoare maximă	Valoare implicită	Wrapper Class
boolean	neprecizat[1)]	-	-	false	**Boolean**
char	16-bit	Unicode 0	Unicode 2^{16}-1	'\x0000 (null)	**Character**
byte	8-bit	-128	+127	(byte)0	**Byte**
short	16-bit	-2^{15}	$+2^{15} – 1$	(short)0	**Short**
int	32-bit	-2^{31}	$+2^{31} – 1$	0	**Integer**
long	64-bit	-2^{63}	$+2^{63} – 1$	0L	**Long**
float	32-bit	IEEE754	IEEE754	0.0f	**Float**
double	64-bit	IEEE754	IEEE754	0.0d	**Double**
void	-	-	-	-	**Void**
-	-	-	-	-	**BigInteger** [2)]
-	-	-	-	-	**BigDecimal** [3)]

Tablourile in Java sunt obiecte

byte x[]; // nealocat, ci doar declarat in **stil C++**

byte [] y; // declarat **stil Java**

- **x si y au valoarea null**

byte v[]=new byte[81]; //alocare + **v.length**

- **v.length are valoarea 81**
- **V[0]=0, v[1]=0, …**

Instante si referinte predefinite

Instanţe predefinite

null – indică faptul că nu s-a atribuit nici o valoare;

- nu poate fi atribuit unei variabile de un tip de date primitiv.
- poate fi folosita pentru a indica disponibilizarea unui obiect

```
Thread fir=new Thread(this);
// ...
fir = null;
```

Referinţe

this – o referire explicită la instanţa curentă

super - o referinţă la superclasă

Pachete si clase

```
package pachet;
public class Punct {// . . .}
```

Referirea claselor:

```
import  pachet.*;
// . . .
Punct m = new Punct();
```

sau

```
pachet.Punct        m  =  new  pachet.Punct();
java.util.Vector    v  =  new  java.util.Vector();
```

DataCalendaristica - NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

<default config>

Projects | Files | Services

- DataCalendaristica — **Proiect**
  - Source Packages
    - datacalendaristica
      - DataCalendaristica.java
  - Libraries
  - jahmm

Navigator

Members | <empty>

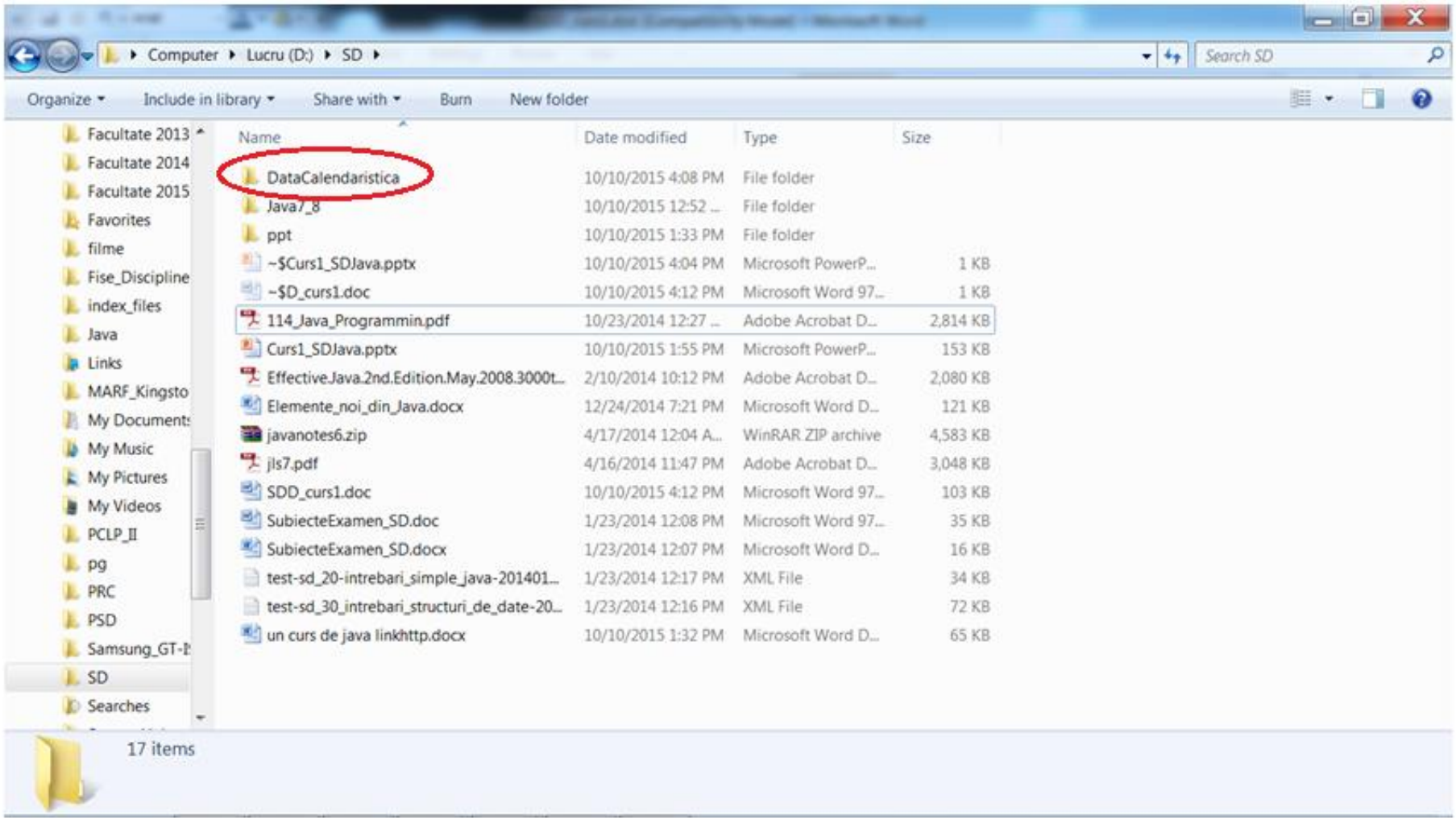
- DataCalendaristica
  - luna : int
  - zi : int
  - an : int
  - main(String[] args)

Start Page | DataCalendaristica.java

Source | History

```
5   */
6   package datacalendaristica;
7
8   public class DataCalendaristica {
9
10      int zi, luna, an=2015;
11      /**
12       * @param args the command line arguments
13       */
14      public static void main(String[] args) {
        zi=10;
        luna=10;
        System.out.println(zi+"/"_luna+"/"+an);
18      }
19
20  }
21
```

Proiect => director



Computer ▸ Lucru (D:) ▸ SD ▸

Search SD

Organize ▾ Include in library ▾ Share with ▾ Burn New folder

Facultate 2013
Facultate 2014
Facultate 2015
Favorites

Name	Date modified	Type	Size
DataCalendaristica	10/10/2015 4:08 PM	File folder	
Java7_8	10/10/2015 12:52 ...	File folder	

Computer ▸ Lucru (D:) ▸ SD ▸ DataCalendaristica ▸

Search DataCalendaristica

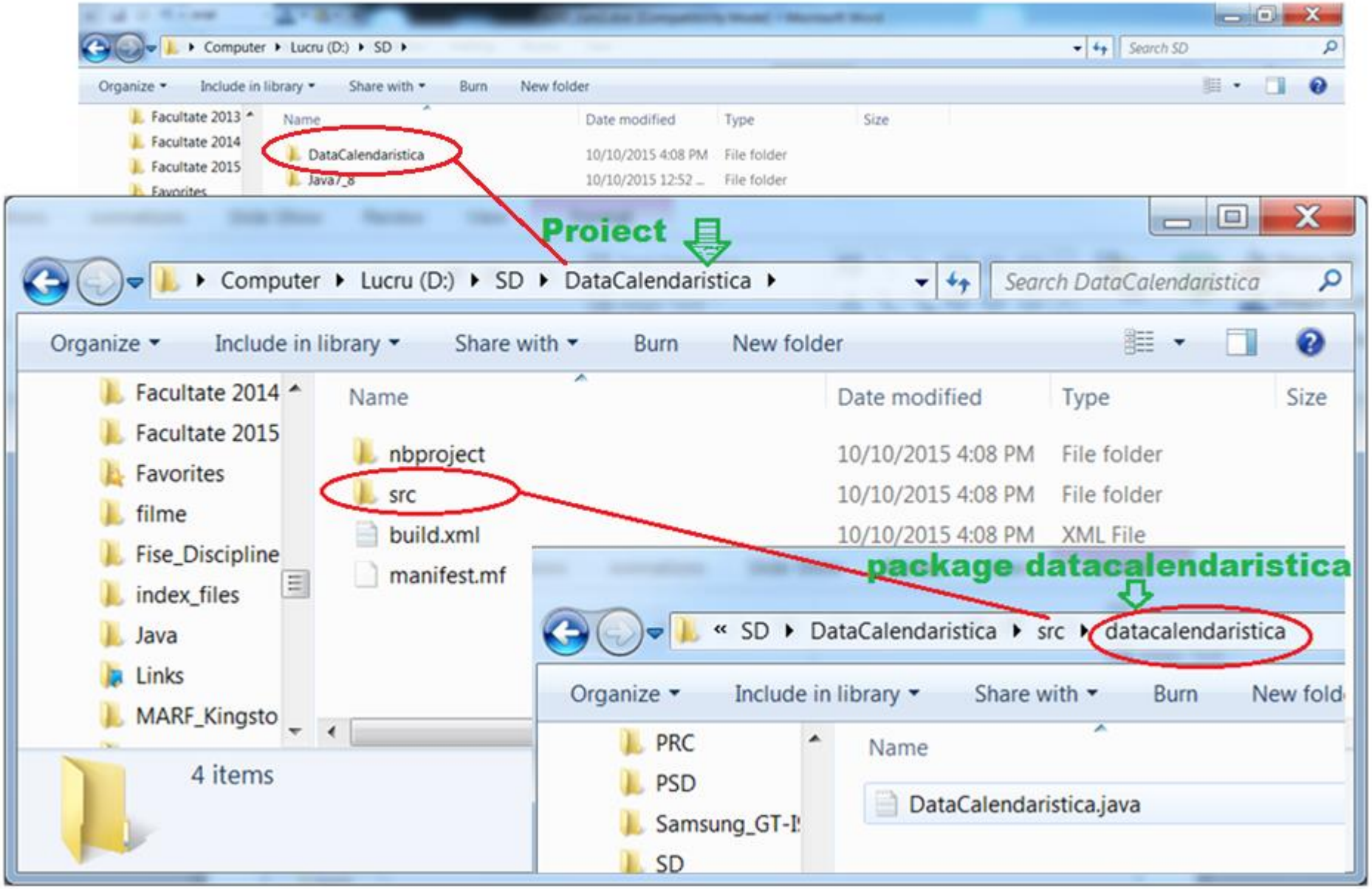
Organize ▾ Include in library ▾ Share with ▾ Burn New folder

Facultate 2014
Facultate 2015
Favorites
filme
Fise_Discipline
index_files
Java
Links
MARF_Kingsto

Name	Date modified	Type	Size
nbproject	10/10/2015 4:08 PM	File folder	
src	10/10/2015 4:08 PM	File folder	
build.xml	10/10/2015 4:08 PM	XML File	
manifest.mf	10/10/2015 4:08 PM	MF File	

4 items

Pachet => director



Erori de compilare

```
package datacalendaristica;

public class DataCalendaristica {
    int zi,luna, an=2015;

    public static void main(String[] args) {
        zi=10;
        luna=10;
        System.out.println ( "Data "+ zi+"/"+ luna + "/“ + an );
 }
}
```

```
11
12
13
14
        zi=10;
```

non-static variable zi cannot be referenced from a static context
---
(Alt-Enter shows hints)

Dupa corectarea erorilor

```
6   package datacalendaristica;
7
8   public class DataCalendaristica {
9
10      int zi, luna, an=2015;
11
12      public static void main(String[] args) {
            DataCalendaristica zicrt= new DataCalendaristica();
14          zicrt.zi=10;
15          zicrt.luna=10;
16          System.out.println("Data "+ zicrt.zi+"/" + zicrt.luna+"/"+zicrt.an);
17      }
18  }
```

datacalendaristica.DataCalendaristica > main > zicrt >

Output - DataCalendaristica (run)

```
run:
Data 10/10/2015
BUILD SUCCESSFUL (total time: 0 seconds)
```

Informatia publica **NU** este o solutie

```
zicrt.zi=37;      // incorect

zicrt.luna=2;
zicrt.zi=30;
// fiecare este corecta considerata separat

zicrt.zi = zicrt.zi+1;
//nu se trece la 1 a lunii urmatoare
```

*Solutia - **încapsulare***

```java
package datacalendaristica;

public class DataCalendaristica {
    private int zi=1, luna=1, an=2015;
    private int ziMax[]= new int[]{31,28,31,30,31,30,31,31,30,31,30,31};

    public int getZi() {
        return zi;
    }
    public void setZi(int zi) {
        if(zi>=1 && zi <=ziMax[ getLuna() - 1])
                                this.zi = zi;
    }

    public int getLuna() {
        return luna;
    }
```

*Solutia - **încapsulare***

```
19      public void setLuna(int luna) {
20          if(luna>=1 && luna <=12) this.luna = luna;
21      }
22
23      public int getAn() {
24          return  an;
25      }
26
27      public static void main(String[] args) {
28          DataCalendaristica zicrt= new DataCalendaristica();
29          zicrt.setZi(15);
30          zicrt.setLuna(10);
31          System.out.println("Data "+ zicrt.getZi()+"/" +
32                                  zicrt.getLuna()+"/"+zicrt.getAn());
33      }
```

*Solutia - **încapsulare***

```
19      public void setLuna(int luna) {
20          if(luna>=1 && luna <=12) this.luna = luna;
21      }
22
23      public int getAn() {
24          return  an;
25      }
26
27      public static void main(String[] args) {
28          DataCalendaristica zicrt= new DataCalendaristica();
29          zicrt.setZi(15);
30          zicrt.setLuna(10);
31          System.out.println("Data "+ zicrt.getZi()+"/" +
32                                  zicrt.getLuna()+"/"+zicrt.getAn());
33      }
```

Output - DataCalendaristica (run)

```
run:
Data crt. 15/10/2015
BUILD SUCCESSFUL (total time: 1 second)
```

*Solutia - **încapsulare***

Avantaje

- Ascunde detaliile de implementare
- Forteaza utilizatorul sa foloseasca o interfata
- Permite asigurarea integrității și coerenței datelor
- Codul devine mai usor de întreținut

Constructor explicit

```
public DataCalendaristica( int _zi, int luna, int an){
    zi = _zi;
    this.luna = luna;
    this.an = an;
}
```

Constructor explicit

```
public DataCalendaristica( int _zi, int luna, int an){
    zi = _zi;
    this.luna = luna;
    this.an = an;
}

public static void main(String[] args) {
    DataCalendaristica zicrt= new DataCalendaristica();
    DataCalendaristica ziNastere = new DataCalendaristica(21,7,1998);

    zicrt.setZi(10);
    zicrt.setLuna(10);
    System.out.println("Data crt. "+ zicrt.getZi()+"/" +
                                zicrt.getLuna()+"/"+zicrt.getAn());
    ziNastere.setZi(20);
    System.out.println("Data nasterii " + ziNastere );
}
```

Cand este prezent un constructor explicit Java nu mai insereaza automat unul implicit!!!

Constructor fără argumente

```
35    public DataCalendaristica( ){
36        java.time.LocalDateTime     data =  java.time.LocalDateTime.now();
37        zi   = data.getDayOfMonth();
38        luna= data.getMonthValue();
39        an   = data.getYear();
40    }
```

Functioneaza în Java 8

Constructor fără argumente

```
35      public DataCalendaristica( ){
36          java.time.LocalDateTime     data =  java.time.LocalDateTime.now();
37          zi   = data.getDayOfMonth();
38          luna= data.getMonthValue();
39          an   = data.getYear();
40      }
41      public static void main(String[] args) {
42          DataCalendaristica zicrt= new DataCalendaristica();
43          DataCalendaristica ziNastere = new DataCalendaristica(21,7,1998);
44
45          System.out.println("Data crt. "+ zicrt.getZi()+"/" +
46                                          zicrt.getLuna()+"/"+zicrt.getAn());
47          ziNastere.setZi(20);
48          System.out.println("Data nasterii " + ziNastere );
49      }
```

Constructor fără argumente

```
35      public DataCalendaristica( ){
36          java.time.LocalDateTime     data =  java.time.LocalDateTime.now();
37          zi   = data.getDayOfMonth();
38          luna= data.getMonthValue();
39          an   = data.getYear();
40      }
41      public static void main(String[] args) {
42          DataCalendaristica zicrt= new DataCalendaristica();
43          DataCalendaristica ziNastere = new DataCalendaristica(21,7,1998);
44
45          System.out.println("Data crt. "+ zicrt.getZi()+"/" +
46                                  zicrt.getLuna()+"/"+zicrt.getAn());
47          ziNastere.setZi(20);
48          System.out.println("Data nasterii " + ziNastere );
49      }
50  }
```

datacalendaristica.DataCalendaristica > DataCalendaristica > data

Output - DataCalendaristica (run)

```
run:
Data crt. 10/10/2015
Data nasterii datacalendaristica.DataCalendaristica@1961c42
BUILD SUCCESSFUL (total time: 1 second)
```

pachet . clasa @ adresa

toString(), @Override

```
@Override
public String toString(){
    return zi +"/" + luna +"/" + an;
}
```

toString(), @Override

```java
    @Override
    public String toString(){
        return zi +"/" + luna +"/" + an;
    }
    public static void main(String[] args) {
        DataCalendaristica zicrt= new DataCalendaristica();
        DataCalendaristica ziNastere = new DataCalendaristica(21,7,1995);

        System.out.println("Data curenta  " + zicrt);
        System.out.println("Data nasterii " + ziNastere);
        ziNastere.setAn( ziNastere.getAn() + 18);
        System.out.println("Majorat       " + ziNastere);
    }
}
```

toString(), @Override

```java
    @Override
    public String toString(){
        return zi +"/" + luna +"/" + an;
    }
    public static void main(String[] args) {
        DataCalendaristica zicrt= new DataCalendaristica();
        DataCalendaristica ziNastere = new DataCalendaristica(21,7,1995);

        System.out.println("Data curenta  " + zicrt);
        System.out.println("Data nasterii " + ziNastere);
        ziNastere.setAn( ziNastere.getAn() + 18);
        System.out.println("Majorat       " + ziNastere);
    }
}
```

datacalendaristica.DataCalendaristica > toString >

ut - DataCalendaristica (run)

```
run:
Data curenta  10/10/2015
Data nasterii 21/7/1995
Majorat       21/7/2013
BUILD SUCCESSFUL (total time: 1 second)
```

Exemplu citire si printf

```
import java.util.Scanner;
class Factorial {
  public static void main(String[] args)  {
    long fact =1;
    int n;
    Scanner kb = new Scanner(System.in);

     System.out.printf(“n=“);
     n =  kb.nextInt();

     for(;n>1; n--)  fact *=n;
     System.out.printf("%d!=%d", n, fact);
   }
}
```

Exemplu citire (String) si printf

```
import java.util.Scanner;
class Factorial {
  public static void main(String[] args)  {
       long fact =1;   int n;
       Scanner kb = new Scanner(System.in);
   try {
     System.out.printf("n=");
     n = Integer.parseInt( kb.nextLine));
     for(;n>1; n--)  fact *=n;
     System.out.printf("%d!=%d", n, fact);
   } catch (FormatNumberException e) {
     System.out.printf("%n%s",
         "Eroare: Nu ati introdus un numar");
   }
  }
}
```

Converters and Flags

Converter	Flag	Explanation
%d		A decimal integer.
%f		A float.
%n		A new line character appropriate to the platform running the application. You should always use %n, rather than \n.
%tB		A date & time conversion—locale-specific full name of month.
%td, %te		A date & time conversion—2-digit day of month. td has leading zeroes as needed, te does not.
%ty, %tY		A date & time conversion—ty = 2-digit year, tY = 4-digit year.
%tl		A date & time conversion—hour in 12-hour clock.
%tM		A date & time conversion—minutes in 2 digits, with leading zeroes as necessary.
%tp		A date & time conversion—locale-specific am/pm (lower case).
%tm		A date & time conversion—months in 2 digits, with leading zeroes as necessary.
%tD		A date & time conversion—date as %tm%td%ty
	08	Eight characters in width, with leading zeroes as necessary.
	+	Includes sign, whether positive or negative.
	,	Includes locale-specific grouping characters.
	-	Left-justified..
	.3	Three places after decimal point.
	10.3	Ten characters in width, right justified, with three places after decimal point.

System.out.format (identic cu printf)

```
long n = 461012;
System.out.format("%d%n", n); // --> "461012"
System.out.format("%08d%n", n); // --> "00461012"
System.out.format("%+8d%n", n); // --> " +461012"
System.out.format("%,8d%n", n); // --> " 461,012"
System.out.format("%+,8d%n%n", n); // --> "+461,012“

double pi = Math.PI;
System.out.format("%f%n", pi); // --> "3.141593"
System.out.format("%.3f%n", pi); // --> "3.142"
System.out.format("%10.3f%n", pi); // --> " 3.142"
System.out.format("%-10.3f%n", pi); // --> "3.142"
System.out.format(Locale.FRANCE, "%-10.4f%n%n", pi);
// -->"3,1416"
```

String.format (*sprintf*)

```
String fs;
float x=14.12536;
int n=10;

fs = String.format(" x= %f%n " + "n=%7d", x, n);

System.out.println(fs);
```

```
x=14.12536
n=      10
```

break *cu eticheta*

```
int[][] mat = {
            { 30, -10, 11},
            { -2, 41, 118 }
};
int val= 118;

cauta:
   for (i = 0; i < mat.length; i++)
     for (j = 0; j < mat[i].length; j++)
        if ( mat [i][j] == val)
                                  break cauta;
```

Instructiunea for - Java 8

```
class VectorIntregi {

  public static void main(String[] args){
        int[] vect = {100,200,300};
         for (int n : vect) {
            System.out.println("N= " + n);
        }
    }
}
```

```
N= 100
N= 200
N= 300
```

Se recomanda utilizarea acestei forme pentru parcurgerea vectorilor.

http://docs.oracle.com/javase/tutorial/java/nutsandbolts/for.html

Moștenire

```
class Parinte {
}

class Copil extends Parinte {
   // moștenește câmpurile și metodele public si protected din clasa Parinte
   //  - daca clasa Copil se gaseste in acelasi pachet cu Parinte mosteneste si câmpurile și
   //     metodele declarate fara un modificator de acces (cel implicit sau package-private)
}
```

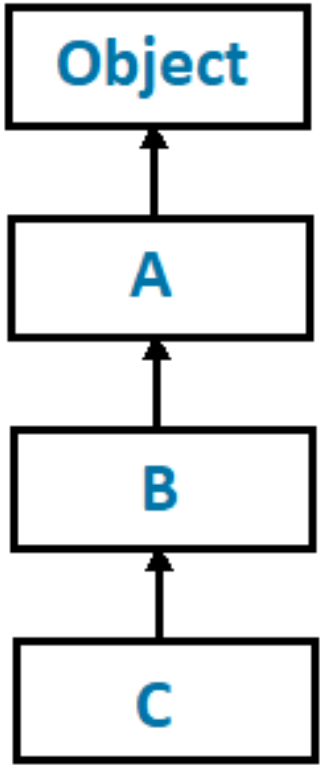
- În Java o clasă poate extinde **o singură clasă** părinte (superclasă)
- Dacă nu este precizat **extends** atunci clasa respectivă extinde clasa **Object** din java.lang

```
class Parinte extends Object { //e redundant extends Object
 }
```

- **Object** este superclasa tuturor claselor Java.

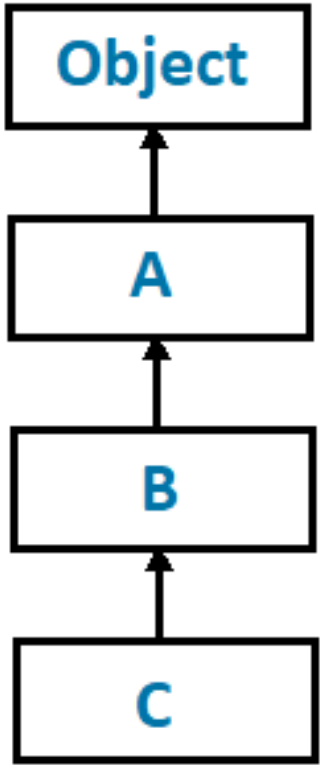
```java
class A {
   int i=1;
   public A() {
      System.out.println("Object()");  //inainte s-a apelat constructorul clasei parinte
      System.out.println("A()  i="+i);
   }
}
class B extends A {
   int j=2;
   public B(){
      System.out.println("B()  j="+j);
   }
}
class C extends B {
   int k=3;
   public C(){
      System.out.println("C()  k="+k);
   }
}

public class Abc {
   public static void main(String[] args) {
      C objc = new C();
      System.out.println("\nobjc.i="+  objc.i+ ",  objc.j="+ objc.j + ",  objc.k=" + objc.k);
    }
}
```



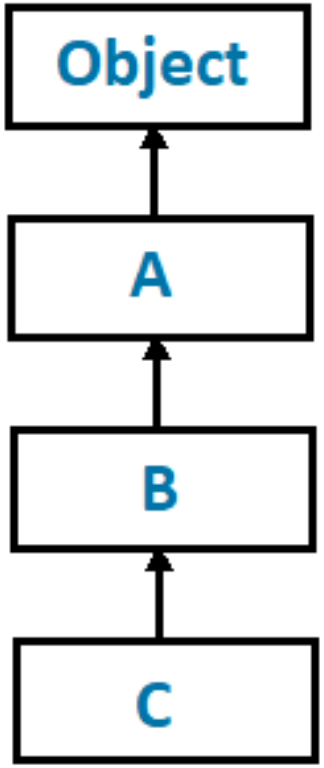
```
class A {
   int i=1;
   public A() {
      System.out.println("Object()");  //inainte s-a apelat constructorul clasei parinte
      System.out.println("A()  i="+i);
   }
}
class B extends A {
   int j=2;
   public B(){
      System.out.println("B()  j="+j);
   }
}
class C extends B {
   int k=3;
   public C(){
      System.out.println("C()  k="+k);
   }
}

public class Abc {
   public static void main(String[] args) {
      C objc = new C();
      System.out.println("objc.i="+  objc.i+
                        ",  objc.j="+ objc.j + ",  objc.k=" + objc.k);
    }
}
```




```
Object()
A()  i=1
B()  j=2
C()  k=3
objc.i=1,  objc.j=2,  objc.k=3
```

```java
class A {
   int i=1;
   public A() {
      super();            //implicit se apeleaza constructorul clasei parinte
      System.out.println("Object()");
      System.out.println("A()  i="+i);
  }
}
class B extends A {
   int j=2;
   public B(){
      super ();           // implicit se apeleaza constructorul clasei parinte
      System.out.println("B()  j="+j);
  }
}
class C extends B {
   int k=3;
   public C(){
      super ();           // implicit se apeleaza constructorul clasei parinte
      System.out.println("C()  k="+k);
  }
}
public class Abc {
   public static void main(String[] args) {
     C objc = new C();
      System.out.println("objc.i="+  objc.i+
                        ",  objc.j="+ objc.j + ",  objc.k=" + objc.k);
   }
}
```




```
Object()
A()  i=1
B()  j=2
C()  k=3
objc.i=1,  objc.j=2,  objc.k=3
```

Inițializarea câmpurilor

Exemple

```
int val;        // val =0, la fel byte, short, long cu 0,  float si double cu 0.0; boolean cu False
Punct p = new Punct (2,3);        // p este o referință nenulă
int x[];          //  variable x might not have been initialized
float[][] y;      // variable y might not have been initialized
float mat[][] = new float[10][];   //mat=[[F@30dae81 mat[9]=null,
                                   // mat[9][0] ->Exception NullPointer…
float [][] w= new float[10][5];    // w=[[F@1b2c6ec2   w[0][0]=0.0
```

Ordinea de inițializare la construirea unui obiect

1. se apelează constructorul clasei părinte
2. se execută inițializările câmpurilor clasei
3. se execută instrucțiunile din constructorul clasei respective

Ordinea de initializarea a campurilor claselor

```
class Initializare {
   public Initializare(int i) {
      System.out.println("Initializare x"+i);
   }
}


class D {
   int i=1
   Initializare x1 = new Initializare(1);
   public D(){
      System.out.println("D()");
      Initializare x2 = new Initializare(2);
   }
   Initializare x3 = new Initializare(3);
}
```

```
class E extends D{
   int j=2;
   Initializare x4 = new Initializare(4);
   public E(){
      Initializare x5 = new Initializare(5);
      System.out.println("B()");
      Initializare x6 = new Initializare(6);
   }
}

public class DemoInitializari {
   Initializare x7 = new Initializare(7);
   public static void main(String[] args)  {
      Initializare x8 = new Initializare(8);
      System.out.println("main: se va executa"+
                                 " E obje = new E();");
      E obje = new E();
      Initializare x9 = newInitializare(9);
      DemoInitializari d = new DemoInitializari();
   }
}
```

Ordinea de initializarea a campurilor claselor

```
public class DemoInitializari {
   Initializare x7 = new Initializare(7);
   public static void main(String[] args)  {
      Initializare x8 = new Initializare(8);
      System.out.println("main: se va executa"+
                                 " E obje = new E();");
      E obje = new E();
      Initializare x9 = newInitializare(9);
      DemoInitializari d = new DemoInitializari();
  }
}

class D {
   int i=1
   Initializare x1 = new Initializare(1);

   public D(){
      System.out.println("D()");
      Initializare x2 = new Initializare(2);
   }

   Initializare x3 = new Initializare(3);
}
```

```
Initializare x8
main: se va executa E obje =
new E();
```

```
Initializare x1
Initializare x3

D()
Initializare x2
```

Ordinea de initializarea a campurilor claselor

```
class E extends D{
   int j=2;
   Initializare x4 = new Initializare(4);
   public E(){
      Initializare x5 = new Initializare(5);
      System.out.println("B()");
      Initializare x6 = new Initializare(6);
   }
}

public class DemoInitializari {
   Initializare x7 = new Initializare(7);
   public static void main(String[] args)  {
      Initializare x8 = new Initializare(8);
      System.out.println("main: se va executa"+
                                 " E obje = new E();");
      E obje = new E();
      Initializare x9 = newInitializare(9);
      DemoInitializari d = new DemoInitializari();
   }
}
```

```
Initializare x4

Initializare x5
B()
Initializare x6

Initializare x9
Initializare x7
```

Constructori și Moștenire

O clasă NU moștenește constructorii de la clasa părinte.

Soluții:

1. se utilizează constructorul implicit (trebuie să existe în clasa părinte)
2. se scriu unul sau mai mulți constructori expliciți

Mostenire

```
public class Employee {
  private static final double BASE_SALARY = 15000.00;
  private String name;
  private double salary;
  private Date   birthDate;

  public Employee(String name, double salary, Date DoB) {
    this.name = name;
    this.salary = salary;
    this.birthDate = DoB;
  }
  public Employee(String name, double salary) {
    this(name, salary, null);
  }
  public Employee(String name, Date DoB) {
    this(name, BASE_SALARY, DoB);
  }
  // more Employee code...
}
```

Mostenire

```
public class Manager extends Employee {
  private String department;

  public Manager(String name, double salary, String dept) {
    super(name, salary);
    department = dept;
  }
  public Manager(String name, String dept) {
    super(name, BASE_SALARY);
    department = dept;
  }
  public Manager(String dept) { // This code fails: no super()
    department = dept;
  }
  //more Manager code...
}
```

Mostenire

```
1   public class Employee extends Object {
2     private String name;
3     private double salary = 15000.00;
4     private Date   birthDate;
5
6     public Employee(String n, Date DoB) {
7       // implicit super();
8       name = n;
9       birthDate = DoB;
10    }
11    public Employee(String n) {
12      this(n, null);
13    }
14  }
```

```
1   public class Manager extends Employee {
2     private String department;
3
4     public Manager(String n, String d) {
5       super(n);
6       department = d;
7     }
8   }
```