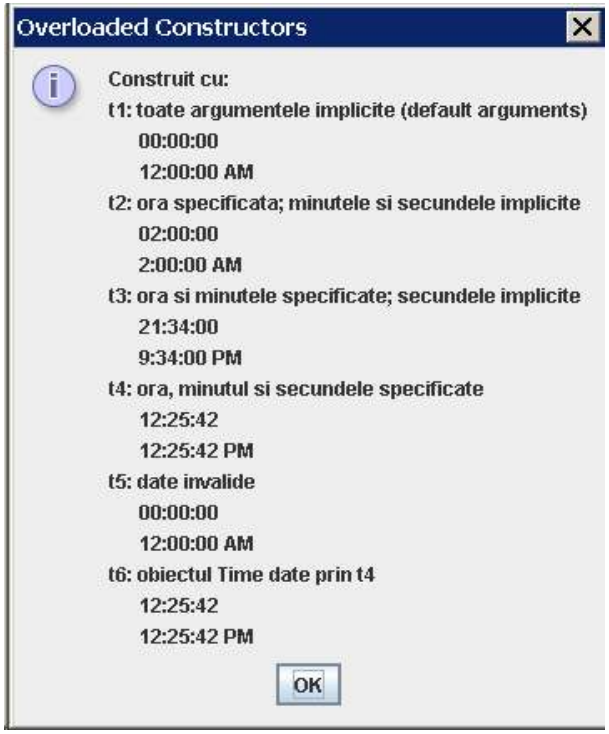
AppOBP > PrOBPT > class Timp

```
import java.text.DecimalFormat;

public class Timp {
   private int hour; // 0 - 23
   private int minute; // 0 - 59
   private int second; // 0 - 59

   // constructorul Timp initializeaza fiecare insanta cu 0;
   // asigura plecarea obiectului Timp dintr-o stare consistenta

   public Timp() {
      this(0, 0, 0); // apel constructor Timp cu 3 argumente
   }

   // Constructor Timp: se da ora, minute si secundele pe 0
   public Timp(int h) {
      this(h, 0, 0); // apel constructor Timp cu 3 argumente
   }

   // Constructor Timp: ora si minutele date, secundele puse pe 0
   public Timp(int h, int m) {
      this(h, m, 0); // apel constructor Timp cu 3 argumente
   }

   // Constructor Timp: ore, minute si secunde date
   public Timp(int h, int m, int s) {
      setTime(h, m, s); // apele setTime pentru validarea timpului
   }

   // Constructor Timp: initializare cu un alt obiect Timp
   public Timp(Timp time) {
      // apel constructor Timp cu 3 argumente
      this(time.hour, time.minute, time.second);
   }

   // setare timp nou cu validarea timpului;
   // valorile invalide se pun pe 0
   public void setTime(int h, int m, int s) {
      hour = ((h >= 0 && h < 24) ? h : 0);
      minute = ((m >= 0 && m < 60) ? m : 0);
      second = ((s >= 0 && s < 60) ? s : 0);
   }

   // conversie la String folosinf formatul de timp universal
   public String toUniversalString() {
      DecimalFormat twoDigits = new DecimalFormat("00");

      return twoDigits.format(hour) + ":" + twoDigits.format(minute) + ":" +
         twoDigits.format(second);
   }

   // conversie la String format de timp standard
   public String toStandardString() {
      DecimalFormat twoDigits = new DecimalFormat("00");

      return ((hour == 12 || hour == 0) ? 12 : hour % 12) + ":" +
         twoDigits.format(minute) + ":" + twoDigits.format(second) +
         (hour < 12 ? " AM" : " PM");
   }
} // end class Timp
```

AppOBP > PrOBPT > class TestTimp

```
//Constructori supraincarcati pt. initializarea obiectelor Timp.
//Overloaded constructors used to initialize Timp objects.
import javax.swing.*;

public class TestTimp {

   public static void main(String[] args) {
      Timp t1 = new Timp(); // 00:00:00
      Timp t2 = new Timp(2); // 02:00:00
      Timp t3 = new Timp(21, 34); // 21:34:00
      Timp t4 = new Timp(12, 25, 42); // 12:25:42
      Timp t5 = new Timp(27, 74, 99); // 00:00:00
      Timp t6 = new Timp(t4); // 12:25:42

      String output =
         "Construit cu: " + "\nt1: toate argumentele implicite (default
arguments)" +
         "\n      " + t1.toUniversalString() + "\n      " +
         t1.toStandardString();

      output +=
            "\nt2: ora specificata; minutele si secundele implicite" + "\n      " +
            t2.toUniversalString() + "\n      " + t2.toStandardString();

      output +=
            "\nt3: ora si minutele specificate; secundele implicite" + "\n      " +
            t3.toUniversalString() + "\n      " + t3.toStandardString();

      output +=
            "\nt4: ora, minutul si secundele specificate" + "\n      " +
t4.toUniversalString() +
            "\n      " + t4.toStandardString();

      output +=
            "\nt5: date invalide" + "\n      " + t5.toUniversalString() + "\n      " +
            t5.toStandardString();

      output +=
            "\nt6: obiectul Time date prin t4" + "\n      " +
t6.toUniversalString() +
            "\n      " + t6.toStandardString();

      JOptionPane.showMessageDialog(null, output, "Overloaded
Constructors",
                                    JOptionPane.INFORMATION_MESSAGE);

      System.exit(0);

   } // end main

} // end class TestTimp
```



AppOBP > PrOBPComp > class Data

```
public class Data {
   private int luna; // 1-12
   private int zi; // 1-31
   private int an; // orice an

   // constructor: apel verificaLuna pt. verificarea lunii;
   // apel verificaZi pt. verificarea zilei

   public Data(int Luna, int Ziua, int Anul) {
      luna = verificaLuna(Luna); // validare luna
      an = Anul; // ar putea fi validat si anul
      zi = verificaZi(Ziua); // validare zi

      System.out.println("Constructor pt. obiectul Data " + toDateString());
   } // end Data constructor

   // metoda de validare a lunii
   private int verificaLuna(int testLuna) {
      if (testLuna > 0 && testLuna <= 12) // validare luna
         return testLuna;

      else { // luna e gresita
         System.out.println("Luna invalida (" + testLuna + ") set to 1.");
         return 1; // metinere obiect in stare consistenta
      }

   } // end metoda verificaLuna

   // metoda de validare a zilei pe baza lunii si a anului
   private int verificaZi(int testZi) {
      int zilePeLuna[] =
      { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

      // verifica daca ziua este in domeniul lunii
      if (testZi > 0 && testZi <= zilePeLuna[luna])
         return testZi;

      // verificare an bisect
      if (luna == 2 && testZi == 29 &&
         (an % 400 == 0 || (an % 4 == 0 && an % 100 != 0)))
         return testZi;

      System.out.println("Zi invalida (" + testZi + ") setata la 1.");

      return 1; // mentinere obiect in stare consistenta

   } // end metoda verificaZi

   // intoarce un String de forma luna/zi/an
   public String toDateString() {
      return luna + "/" + zi + "/" + an;
   }

} // end class Date
```

AppOBP > PrOBPComp > class Angajat

```
public class Angajat {
   private String nume;
   private String prenume;
   private Data dataNastere;
   private Data dataAngajare;
   private long salar;

   // constructor initializare nume data nasterii si angajare
```

```
   public Angajat(String first, String last, Data dataDeNastere,
                  Data dataAngajarii, long salarFirma) {
      nume = first;
      prenume = last;
      dataNastere = dataDeNastere;
      dataAngajare = dataAngajarii;
      salar = salarFirma;
   }

   // conversie Angajat la String
   public String toEmployeeString() {
      return prenume + ", " + nume + "  Angajat la: " +
         dataAngajare.toDateString() + "  Zi de nastere: " +
         dataNastere.toDateString() + " Salar: " + salar;
   }

   public void cresteSalar(double cuProcent) {
      double crestere = salar * cuProcent / 100.;
      salar += crestere;
   }

} // end class Angajat
```

AppOBP >
PrOBPComp > class TestAngajat

```
import javax.swing.JOptionPane;
public class TestAngajat {
   public static void main(String[] args) {
      Data birth = new Data(7, 24, 1949);
      Data hire = new Data(3, 12, 1988);

      Angajat[] angajati = new Angajat[3];
      Angajat a1;

      a1 = new Angajat("Vasile", "Ion", birth, hire, 19000);
      angajati[0] = new Angajat("Nelu", "Sile", birth, hire, 19000);
      angajati[1] = new Angajat("Ada", "Milena", birth, hire, 19000);
      angajati[2] = new Angajat("Ale", "Ion", birth, hire, 19000);

      //afisare a1
      JOptionPane.showMessageDialog(null, a1.toEmployeeString(),
                        "Tesare clasa Angajat",
                        JOptionPane.INFORMATION_MESSAGE);

      //afisare angajati
      for (Angajat a : angajati)
         System.out.println(a.toEmployeeString());

      //crestem salarul cu 5%
      for (Angajat a : angajati)
         a.cresteSalar(5);

      //si asa se poate pargurge tabloul
      for (int i = 0; i < angajati.length; ++i)
         System.out.println(angajati[i].toEmployeeString());

      System.exit(0);
   }
} // end class TestAngajat
```

Constructor pt. obiectul Data 7/24/1949
Constructor pt. obiectul Data 3/12/1988
Sile, Nelu Angajat la: 3/12/1988 Zi de nastere: 7/24/1949 Salar: 19000
Milena, Ada Angajat la: 3/12/1988 Zi de nastere: 7/24/1949 Salar: 19000
Ion, Ale Angajat la: 3/12/1988 Zi de nastere: 7/24/1949 Salar: 19000
Sile, Nelu Angajat la: 3/12/1988 Zi de nastere: 7/24/1949 Salar: 19950
Milena, Ada Angajat la: 3/12/1988 Zi de nastere: 7/24/1949 Salar: 19950
Ion, Ale Angajat la: 3/12/1988 Zi de nastere: 7/24/1949 Salar: 19950