Clasa Scanner

Scrieri cu format*

Exerciţii şi probleme

Clasa Scanner

Exemple

1. În următorul exemplu sunt citite două numere întregi de la tastatură şi se afişează suma lor

```
import java.util.Scanner;
class ExpScanner{
     public static void main(String arg[]){
          Scanner sc=new Scanner(System.in);
          System.out.print("Introduceti doua numere intregi: ");
          int x=sc.nextInt();
          int y=sc.nextInt();
          System.out.println("Suma lor este "+(x+y));
     }
}
```

2. Să presupunem că dorim să citim un întreg. În caz că valoarea introdusă de utilizator nu reprezintă un întreg, îi permitem utilizatorului să reintroducă valoarea. Utilizatorul va putea reintroduce valoarea de un număr maxim de ori.

```
import java.util.Scanner;
class TestScan{
  static final int NR_MAX=3;
  public static void main(String arg[]){
    int i,k=0;
    Scanner sc=new Scanner(System.in);
    System.out.print("Introduceti un intreg: ");
    i=0;
    while(i<NR_MAX)
      if(sc.hasNextInt()){
          k=sc.nextInt();
          i=NR_MAX;
      }
      else{
          String s=sc.next();
          System.out.print(s+" nu este o valoare intreaga.");
          i++;
          if(i<NR_MAX)
              System.out.print("Incercati din nou: ");
          else
              System.exit(1);
      }
     System.out.printf("Ati introdus intregul %d",k);
  }
}
```

3. Având o clasă `Muchie` cu câmpurile reprezentând vârful inițial, vârful final şi costul muchiei şi un fişier cu următoarea structură: pe prima linie avem numărul de muchii, iar pe următoarele câte trei numere reprezentând vârful inițial, vârful final şi costul unei muchii, ca de exemplu:

```
5
2 3 10.5
1 3 4
1 2 17.23
1 4 20
3 4 0.5
```

să se construiască un vector de muchii cu datele citite din fişier.

```
import java.util.Scanner;
import java.util.InputMismatchException;
import java.util.NoSuchElementException;
//import java.util.Locale;
import java.io.FileNotFoundException;

class Muchie{
      int vi,vf;
      double cost;
      Muchie(int vi,int vf, double cost){
            this.vi=vi;
            this.vf=vf;
            this.cost=cost;
      }
      public String toString(){
            return "("+vi+","+vf+")"+" "+cost;
      }
}

class TestScanF{
  public static void main(String arg[]){
    int i,n;
    Muchie muchii[];
    try{
      Scanner scFisier=new Scanner(new java.io.File("muchii.in"));
      //scFisier.useLocale(Locale.ENGLISH);
      n=scFisier.nextInt();
      muchii=new Muchie[n];
      for(i=0;i<n;i++)
         muchii[i]=new Muchie(scFisier.nextInt(), scFisier.nextInt(),
                              scFisier.nextDouble());
      scFisier.close();
      for(i=0;i<n;i++)
         System.out.println(muchii[i]);
    }
```

```
    catch(FileNotFoundException fnf){
        System.out.println("Fisier inexistent.");
    }
    catch(InputMismatchException im){
        System.out.println("Date de tip incorect.");
    }
    catch(NoSuchElementException nse){
     System.out.println("Nu exista informatii despre toate muchiile.");
    }
  }
}
```

ⓘ **Observaţii**:

1. Nu este obligatoriu să tratăm erorile `InputMismatchException` şi `NoSuchElementException` aruncate de metodele `nextInt()` şi `nextDouble()` . În acest caz puteam folosi metodele `hasNextInt()` sau `hasNextDouble()` ale clasei `Scanner` dacă vroiam să ne asigurăm ca datele citite sunt corecte.
2. Este obligatoriu să tratăm eroarea `FileNotFoundException`, altfel vom avea eroare la compilare

Scrieri cu format*

Pentru scrieri cu format se poate folosi o formă a metodei `printf` a clasei `PrintStream` (`System.out` este un obiect de tip `PrintStream`).

```
printf(String format, Object ... args)
```

Parametrul format este un şir de caractere care conţine cel puţin câte o specificare a formatului pentru fiecare argument care urmează. Această specificaţie are forma generală

```
%[argument_index$][flags][width][.precision]conversion
```

(parantezele drepte au semnificaţia de opţional)

Opţiunile din forma generală pot lua următoarele valori:

`conversion` – un caracter ce specifică tipul, de exemplu:

- ‘d’, ‘o’ şi ‘x’ se folosesc pentru întregi şi arată că reprezentarea valori este în baza 10, 8, respectiv16 .

- ‘f’, ‘g’ şi ‘a’ se folosesc pentru valori reale şi arată că reprezentarea va fi în notaţie zecimală, ştiinţifică sau hexazecimală cu exponent.
- ‘c’ pentru caracter
- ‘s’ pentru şir de caracter
- ‘b’ pentru boolean (afişat true sau false)

`argument_index` – un întreg reprezentând numărul argumentului care urmează parametrului de format. De exemplu “`1$`” se referă la primul argument. Se poate folosi şi simbolul ‘<’ in loc de o secvenţă de tipul `număr$`, indicând faptul că se foloseşte acelaşi argument ca şi la specificarea anterioară.

`flags` – este un set de caractere care modifică formatul de ieşire. De exemplu

- ‘+‘ forţează scrierea semnului pentru valori numerice
- ‘0‘ forţează completarea valorilor numerice cu 0
- ‘-‘ arată că argumentul va fi aliniat la stânga

`width` – numărul minim de caractere pe care va fi scrisă ieşirea
`precision` – folosit de exemplu pentru a specifica numărul de zecimale pentru valori reale.

Exemplu

```
class ExpPrintf{
      public static void main(String s[]){
            String s1="ab";
            int x=3;
            float f1=1.2345f,f2=3f;
            System.out.printf("Un intreg %d",x);
            System.out.println();
            System.out.printf("Un intreg scris pe sase caractere
aliniat la dreapta %6d",x);
            System.out.println();
            System.out.printf("Numarul PI cu 5 zecimale %.5f",Math.PI);
            System.out.println();
            System.out.printf("Numarul PI cu 5 zecimale scris pe zece
caractere %10.5f",Math.PI);
            System.out.println();
            System.out.printf("primul numar real %f,al doilea numar %f
\nal doilea numar cu semn %<+f, primul numar cu 2 zecimale
%1$.2f\n",f1,f2);
            System.out.printf("Sir aliniat la dreapta %10s, la stanga
%1$-10s sirul %1$s",s1);
      }
}
```

Exerciţii şi probleme

🕮 Exerciţii:

1. Să se creeze o clasă `MatriceTriunghiulara` care are două câmpuri: un număr întreg `n` şi un tablou bidimensional `a` de numere întregi, precum şi două metode:
- o metodă pentru citirea matricei inferior triunghiulară `a` de dimensiune `n`
- o metodă de afişare a acestei matrice.
Să se scrie o clasă principală în care să se creeze şi să se afişeze o matrice inferior triunghiulară, folosind clasa `MatriceTriunghiulara`

De exemplu:
```
n=4
a=
3
4 7
9 0 6
1 5 8 2
```
(matricea se va afişa sub această formă triunghiulară)

2. Creați un tablou de numere complexe (de tipul `Complex` implementat anterior).
a) Afișați numerele din tablou și modulul fiecăruia folosind instrucțiunea for pentru colecții
b) Adunați la primul număr din tabloul obținut celelalte numere și afișați numărul rezultat.

3. Se citesc de la tastatură două şiruri de caractere. Să se determine numărul de apariţii ale primului şir în cel de al doilea.

4. Scrieți o aplicație Java care să afișeze argumentele primite în ordine lexicografică.

5. Scrieți o aplicație Java care împarte o propoziție primită de la tastatură în cuvinte (cuvintele se pot separa prin spațiu, virgulă sau punct).

6. Valoarea unui şir de caractere de tip **`String` nu poate fi modificată după creare** (de exemplu nu se poate modifica un caracter al șirului). Dacă este necesară și modificarea șirului de caractere se pot utiliza clasele `StringBuilder` sau `StringBuffer`. Consultaţi documentaţia şi scrieţi un exemplu în care să folosiţi metode ale clasei `StringBuilder` care modifică valoarea şirului curent.

✍ Problemă

Scrieți o clasă pentru o listă simplu înlănțuită.

Folosiți această clasă pentru a parcurge în lățime un graf neorientat reprezentat prin liste de adiacență (clasa scrisă pentru listă se va folosi la reprezentarea grafului și pentru coada necesară la parcurgerea în lățime). Datele se vor citi din fișier.