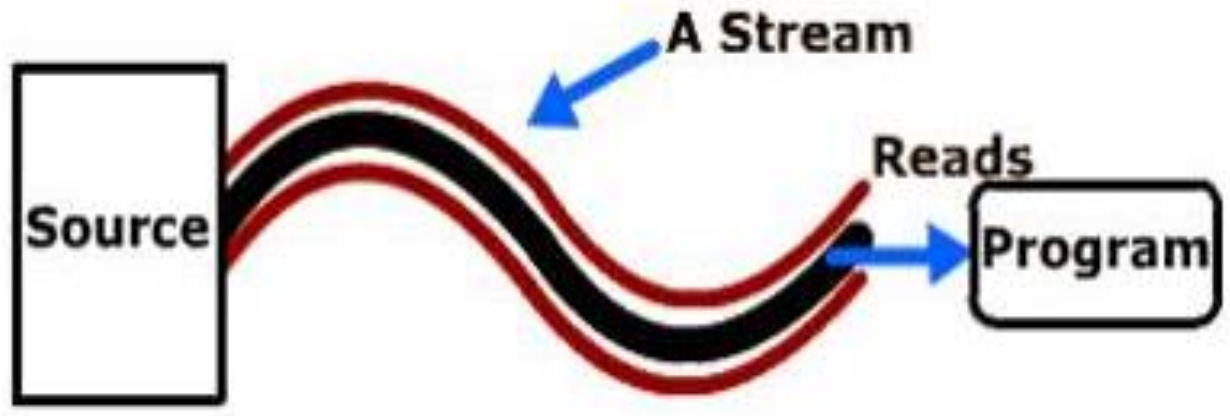
Java

IO műveletek

I/O (input/output)

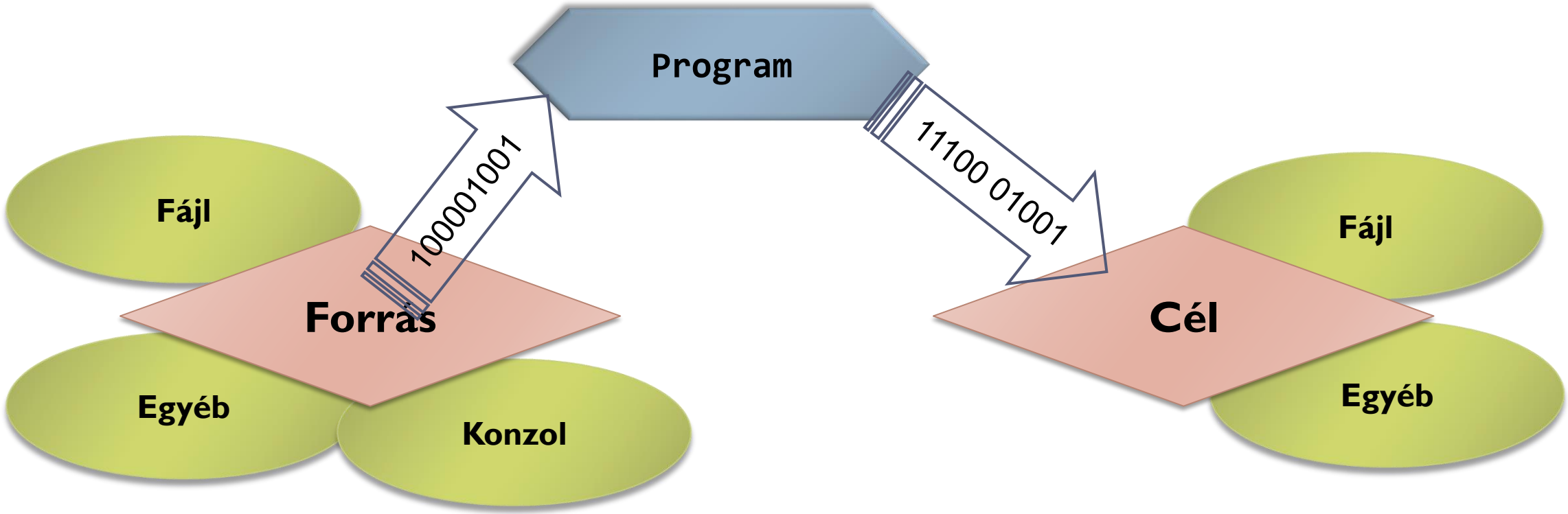
Az összes olyan folyamatot, ami a program és a külvilág közötti kommunikációért felelős I/O (input/output) műveleteknek nevezzük.

Ezt a kommunikációt a Java adatfolyamokon, más néven **Stream**-eken keresztül valósítja meg.



Az adatfolyamok nagy részét úgy kell elképzelni, mint egy csövet egy csappal, amelyet meg kell nyitni ahhoz, hogy áthaladhasson rajta az, amit szállít.

I/O (input/output)



Konzol

Eddig kizárólag konzolon keresztül kommunikáltunk a programunkkal .

A konzol kezelésére a Java három olyan adatfolyamot biztosít, melyeket nem kell nyitni-zárni ahhoz, hogy kommunikálhassunk rajta keresztül, ezek a Standard Stream-ek.

Konzol

Standard kimenet: System.out

onnan lehet ismerős, hogy a kezdetektől ezt használtuk kiíratásra, vagyis már akkor is Stream-et használtunk.

Standard bemenet: System.in

*a **Scanner** osztálynak ezt kellett odaadni, ez alapértelmezetten a billentyűzetet jelenti a konzolban*..

Standard hiba: System.err

ez is gyakorlatilag egy olyan kiíratást végez el, mint a System.out, de ezt csak hibaüzenet kiíratásra szokás használni.

Konzol

```
java.util.Scanner sc = new java.util.Scanner(System.in);
System.out.print("  a = "); a = sc.nextInt();

System.out.println("  1. feladat ");

System.out.println(  String.format(”%10s %3.2f”,”Béla”, x) );
```

Fájlkezelés

A Java nyelvben a fájlkezelés Stream-eken keresztül valósul meg.

Ezeket azonban csak akkor használhatjuk, ha a programunk elején importáljuk a java.io osztályt, mely ezeket a Stream-eket tartalmazza.

A programunk elejét tehát kezdjük ezzel:

```
import java.io.*;
```

Fájlkezelés

import java.io.* nem csak a Stream-ek használatához szükséges kódokat tartalmazza, hanem a hibakezelés megfelelő osztályait is.

Pl.:

- nincs meg a fájl,
- a helyét adtuk meg rosszul,
- a nevét írtuk el,
- nem működik az adathordozó,

Mindenképpen egy kivételkezelő szerkezettel kell megoldani.

Fájlkezelés

Szintaxis:

```
try{

        // megpróbáljuk (try) beolvasni, feldolgozni a fájlt

}catch(IOException e){

        // a hiba elfogása (catch) – I/O kivétel kezelése

}
```

Fájlkezelés

```
import java.io.*;
  String s;
  RandomAccesFile f;

  try{
    f=new RandomAccesFile(”adatok.txt”,”r”);
     // feldolgozás
    f.close();
  }catch(IOException e){
     System.out.println(”Baj van”);
  }
```

Fájlkezelés

A beolvasott állományt valahol tárolni kell. Hogyan? Soronként? A sorokat még tovább bonthatjuk?

- A fájl sorait minden esetben mint karakterláncokat olvassuk be.
- Ha ezek számokat tartalmaznak, azokat át kell majd alakítanunk.
- Ha csak a nyers beolvasott sorokat akarjuk tárolni, akkor ehhez egy String tömbre van szükségünk.

Fájlkezelés

A beolvasási problémák:

1. Előre tudjuk, hány sorból áll a fájl
2. Nem tudjuk, hány sorból áll a fájl, de az első sorban megtaláljuk a sorok darabszámát
3. Nem tudjuk, hány sorból áll a fájl

```
package javaio_01;
import java.io.*;
static void f1() {  //tudjuk,hogy 10 sorból áll a fájl
    String sor; int N=10; String nevek[]=new String[N];
    try {
        f = new RandomAccessFile ("nevek.txt", "r");
        for (int i=0;i<N;i++){
                sor = f.readLine(); nevek[i] = sor;
         }
         f.close();
    } catch(IOException e) {
          System.out.println(”Hiba”); }
} //f1 vége
```