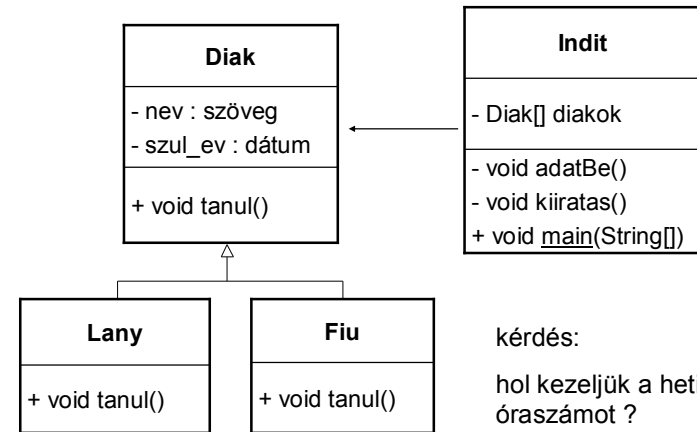


## Programozás III

OOP ÖRÖKLŐDÉS,  
PÉLDÁK

### JAVA ÖRÖKLŐDÉS – PÉLDA



### JAVA ÖRÖKLŐDÉS – PÉLDA

Példa:

A diákokat jellemzi a nevük és születési dátumuk. Minden diák tanul, de a lányok és fiúk nem teljesen azonos módon: a lányok alaposabban, a fiúk nagyvonalúan.

Minden lány ugyanannyi órát tanul hetente, és minden fiú is ugyanannyit, de a fiúk, lányok óraszámuk különbözik.

Hozunk létre egy diákokat tartalmazó tömböt, és írassuk ki a benne szereplők nevét, tanulási módját.

Megjegyzés: ez a példa – bár kissé(?) béna, de alkalmas az öröklődés és a polimorfizmus szemléltetésére is.

### JAVA ÖRÖKLŐDÉS – PÉLDA

```
public class Diak{
    private String nev;
    private int szul_ev;
    private static int ora;

    public Diak(String nev, int szul){
        this.nev = nev;
        this.szul_ev = szul;
    }

    public void tanul(){
        System.out.println(nev + " " + ora + " órát tanul hetente.");
    }

    public static void setOra(int ora){
        Diak.ora = ora;
    }
}
```

#### Figyelem!

Alapsztályban NINCS kiíratás, most kizárólag az egyszerűség kedvéért szerepel így.

### JAVA ÖRÖKLŐDÉS – PÉLDA

```
public class Fiu extends Diak{  
  
    public Fiu(String nev, int szul){  
        super(nev,szul);  
    }  
  
    public void tanul(){  
        super.tanul();  
        System.out.println("nagyvonalúan tanul");  
    }  
}  
  
public class Lany extends Diak{  
  
    public Lany(String nev, int szul){  
        super(nev,szul);  
    }  
  
    public void tanul(){  
        super.tanul();  
        System.out.println("alaposan tanul");  
    }  
}
```

### JAVA ÖRÖKLŐDÉS – JAVÍTOTT PÉLDA

```
public class Diak{  
  
    private String nev;  
    private int szul_ev;  
  
    public Diak(String nev, int szul){  
        this.nev = nev;  
        this.szul_ev = szul;  
    }  
  
    public void tanul(){  
        System.out.print(nev + " heti tanulási ideje: ");  
    }  
}
```

### JAVA ÖRÖKLŐDÉS – PÉLDA

```
System.out.print("Lányok tanulási óraszám: ");  
Lany.setOra(Input.readInt()); 5  
System.out.print("Fiúk tanulási óraszám: ");  
Fiu.setOra(Input.readInt()); 3
```

Kérdés: ki hány órát tanul?      Ádám 3 órát tanul hetente.  
nagyvonalúan tanul  
Éva 3 órát tanul hetente.  
alaposan tanul

Tanulság:  
Osztályváltozó/osztálymetódus örökítésével **NAGYON**  
csínján kell bánni!

### JAVA ÖRÖKLŐDÉS – JAVÍTOTT PÉLDA

```
public class Fiu extends Diak{  
  
    private static int ora;  
  
    public Fiu(String nev, int szul){  
        super(nev,szul);  
    }  
  
    public void tanul(){  
        super.tanul();  
        System.out.println(ora + "\nagyvonalúan tanul");  
    }  
  
    public static void setOra(int ora){  
        Fiu.ora = ora;  
    }  
}
```

A Lany ugyanilyen, csak a tanul() szövege más.

## JAVA ÖRÖKLŐDÉS – TOVÁBBI KÉRDÉS

A Fiu osztályban hozzunk létre egy focizik() metódust:

```
public void focizik(){
    //...
}
```

Hívjuk meg:

```
Diak diak = new Fiu(nev, szulEv);
diak.
  equals(Object o)    boolean
  getClass()         Class<?>
  getNev()           String
```

## JAVA ÖRÖKLŐDÉS – TOVÁBBI KÉRDÉS

Helyesen:

```
if (diak instanceof Fiu) {
    ((Fiu) diak).focizik();
}
```

Még helyesebben:

Ha csak lehet, kerüljük a típuskényszerítést!

## JAVA ÖRÖKLŐDÉS – TOVÁBBI KÉRDÉS

Típuskényszerítés kell:

```
((Fiu)diak).
  equals(Object o)    boolean
  focizik()           void
```

Formája:

**((Típus)peldany)** - fontos a zárójelezés!

Mi történik, ha a diak példány történetesen Lany? ☹️

## JAVA ÖRÖKLŐDÉS – ISMÉTLÉS

Módosítók használata:

Adattag: private

Lehet-e protected? nem, mert nem biztonságos

Lehet-e public? csak akkor, ha final

Metódus: private, protected, public vagy módosító nélküli

Konstruktorból hívott metódus legyen private. (De csak inicializáló metódust hívunk!)