

Egységbezárás / Adatrejtés



Gyakorlat:

Tort, Kor,

MindentudoGep

Pont, Teglalap, Vonal

C++, C#, *Java*

Egységbezárás – class készítése

- ▶ Az azonos fogalomhoz tartozó „dolgokat” egy egységbe zárjuk. Az egységünk lesz az Osztály.
- ▶ Az Osztály egy típus, melyből példányosítás (new) során objektumot hozunk létre.
- ▶ Az objektum egységbe zárja az adattagjait (korábban változók, ez mutatja az obj. állapotát), és viselkedését (ezek a tagfüggvények, korábban metódusok).

Koordináta tárolása

- ▶ Eddigi lehetőségek alaptípusokkal:

struct **Összetett típusok**

```
Koord{  
    int x;  
    int y;  
};
```

- ▶ Használat:

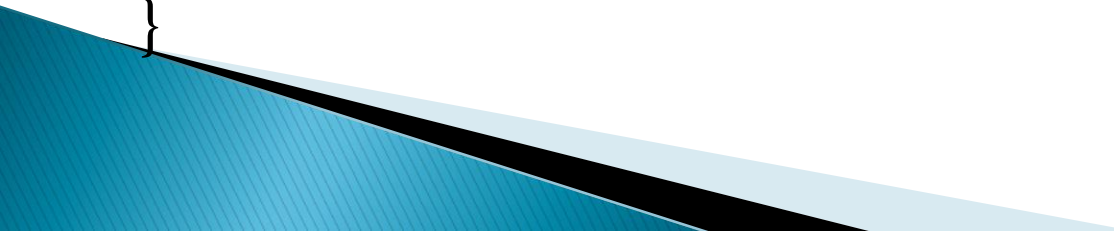
	<u>Típus</u>	<u>azonosító</u>
alap	→ int	szam;
összetett	→ Koord	koord;
	→ Random	rnd;

```
struct Koord{  
    int x;  
    int y;  
};
```

```
Koord koord;  
koord.x = 20;  
koord.y = 30;
```

Tört készítése, tárolása

```
int main(){  
    struct tort{  
        int szamlalo;  
        int nevező;  
    };  
    tort t;  
    t.nevező = 1;  
    t.szamlalo = 3;  
  
    cout << "tört értéke:" << t.szamlalo << "\\\" << t.nevező;  
}
```

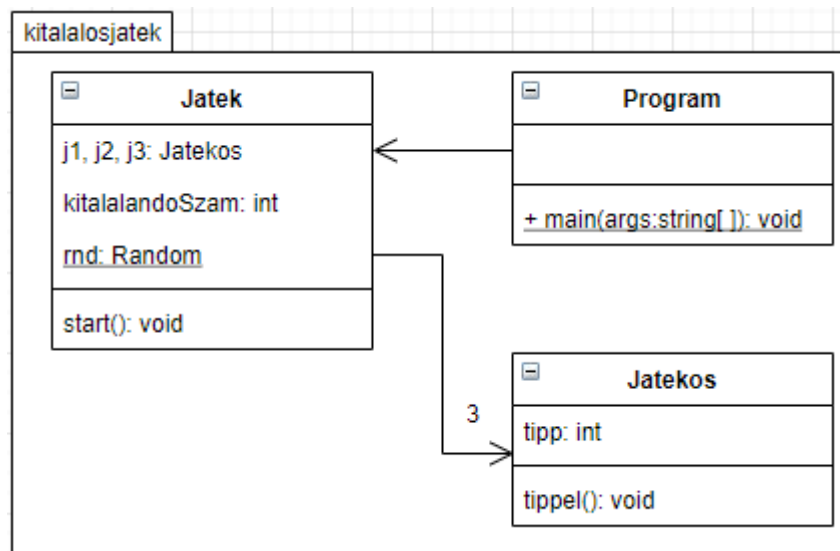


Feladatok

1. Alakítsd át a KitalalosJatekot az UML-nek megfelelően
2. Alakítsd át a TortProgramot az UML-nek megfelelően
3. Készítsd el a KorProgramot:
 - Milyen szereplő van a programban
 - Mi a jellemző (adattagok) erre a szereplőre
 - Ezekkel a jellemzőkkel mit lehet csinálni (tagfüggvények)

Kitalálós Játék UML osztálydiagram

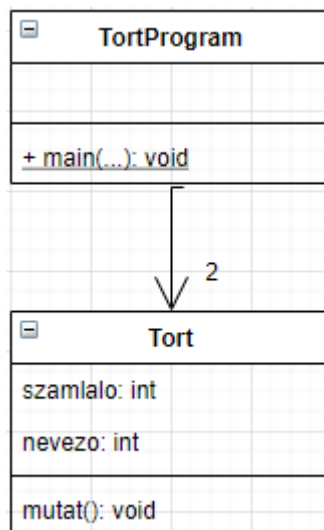
A kész programot alakítsd át az UML-nek megfelelően!



A main is és az rnd is alá van húzva! Ez jelzi a static kulcsszót!

Tört készítése, tárolása

```
int main(){
    struct tort{
        int szamlalo;
        int nevező;
    };
    tort t;
    t.nevező = 1;
    t.szamlalo = 3;
```



```
package tortprogram;
public class TortProgram {
    public static void main(String[] args) {
        Tort tort1;
        tort1 = new Tort();
        tort1.szamlalo = 1;
        tort1.nevező = 3;
        tort1.mutat();

        Tort tort2 = new Tort();
        tort2.szamlalo = 3;
        tort2.nevező = 5;
        tort2.mutat();
    }
}
```

```
package tortprogram;
//Egységbezáras
public class Tort {
    int szamlalo, nevező;
    void mutat() {
        System.out.println("tört értéke: "
            + szamlalo + "/" + nevező);
    }
}
```

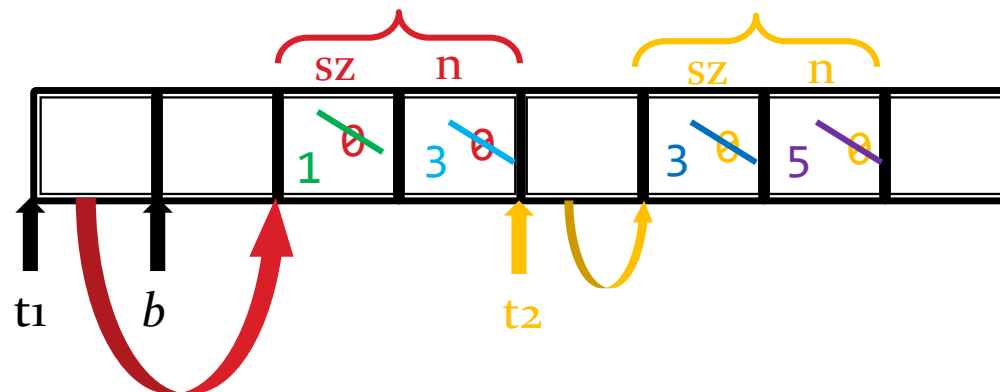
```
cout << "tört értéke:" << t.szamlalo << "\\\" << t.nevező;
```

```
}
```

Tort a memóriában

➔ Tort t1;
 ➔ char b = 'P';
 ➔ t1 = new Tort();
 ➔ t1.szamlalo = 1;
 ➔ t1.nevezo = 3;

➔ Tort t2 = new Tort();
 ➔ t2.szamlalo = 3;
 ➔ t2.nevezo = 5;



```

public class Tort {
    int szamlalo; //SZ
    int nevezo; //n
}
    
```


TortProgram – további lehetőségek

► Mit tud még egy tört?

- Kiszámolja az értékét (pl. melyik nagyobb?)
- Reciprokra vált (változik az állapota!)
- Visszaadja a reciprok értékét (ez is egy tört lesz!)

```
double ertek(){  
    return szamlalo / nevező;  
}
```

```
void reciprokraValt(){  
    int csere = nevező;  
    nevező = szamlalo;  
    szamlalo = csere;
```

```
}
```

```
Tort reciprokotVisszaAd(){  
    Tort t = new Tort();  
    t.szamlalo = this.nevező;  
    t.nevező = this.szamlalo;  
    return t;
```

```
}
```

KorProgram

Jellemzők (adattagok)

- ▶ Sugár
- ▶ Pozíció (x, y)
- ▶ Vonal
 - Vastagság
 - Szín
 - Típus

Lehetőségek (tagfüggvények)

- ▶ terület(), kerület(), nyujt()
- ▶ mozgat()
- ▶ getSzin(), setSzin(String szin)