

## 6. Kapcsolatok modellezése és megvalósítása

Határidő márc 25, 23:59

Pont 10

Kérdések 10

Időkorlát Nincs

Engedélyezett próbálkozások 5

[Kvíz kitöltése újra](#)

### Próbálkozások naplója

	Próbálkozás	Idő	Eredmény
LEGUTOLSÓ	<a href="#">1. próbálkozás</a>	2 perc	2.33 az összesen elérhető 10 pontból

⚠ A helyes válaszok el vannak rejtve.

Ezen próbálkozás eredménye: **2.33** az összesen elérhető 10 pontból

Beadva ekkor: márc 25, 15:30

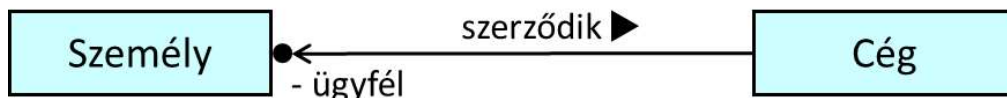
Ez a próbálkozás ennyi időt vett igénybe: 2 perc

Helytelen

#### 1. kérdés

0 / 1 pont

Egy biztosításokkal foglalkozó cég és egy személy között akkor jön létre kapcsolat, amikor a személy biztosítást köt a céggel.



Mi a módja e kapcsolat felépítésének?

☐

A Cég osztály egy metódusa kapja meg paraméterként a szerződést kötő személy hivatkozását, amit az *ügyfél* nevű adattagjában tárol el.

☐

A Személy osztály konstruktora kapja meg paraméterként a cég hivatkozását, amit az *ügynök* *ügyfél* nevű adattagjában tárol el.



A *Cég* osztály konstruktora kapja meg paraméterként a szerződést kötő személy hivatkozását, amit az *ügyfél* nevű adattagjában tárol el.

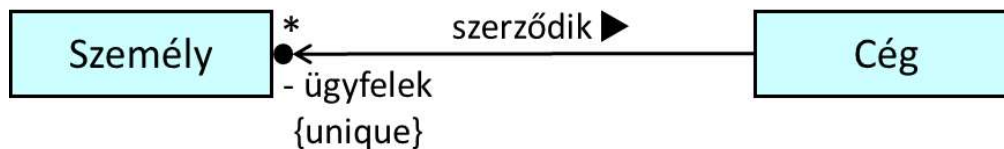


A *Személy* osztály egy metódusa kapja meg paraméterként a cég hivatkozását, amit a saját *ügyfél* nevű adattagjában tárol el.

## 2. kérdés

1 / 1 pont

Egy biztosításokkal foglalkozó cég és egy személy között akkor jön létre kapcsolat, amikor a személy biztosítást köt a céggel. Ilyenkor a cégnek már lehetnek korábban szerződött ügyfelei.



Mi a módja egy újabb személlyel történő kapcsolat felépítésének?



A *Személy* osztály egy metódusa kapja meg paraméterként a cég hivatkozását, amelyet hozzáad az *ügyfelek* nevű gyűjteményéhez, ha még nem szerepel benne.



A *Személy* osztály egy metódusa kapja meg paraméterként a cég hivatkozását, amelyet hozzáad az *ügyfelek* nevű gyűjteményéhez.



A *Cég* osztály egy metódusa kapja meg paraméterként az ügyfél hivatkozását, amit ez a metódus hozzáad a cég *ügyfelek* nevű gyűjteményben tárolt ügyfelekhez



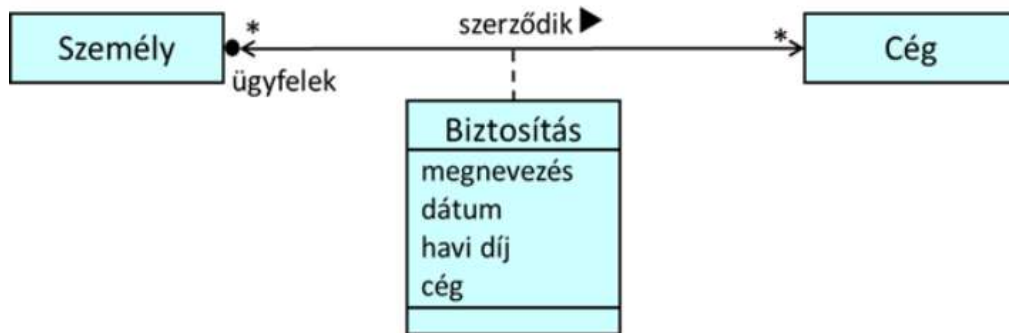
A *Cég* osztály egy metódusa kapja meg paraméterként az ügyfél hivatkozását, amit ez a metódus hozzá ad az *ügyfelek* nevű gyűjteményéhez, ha még nem szerepel benne.

Helytelen

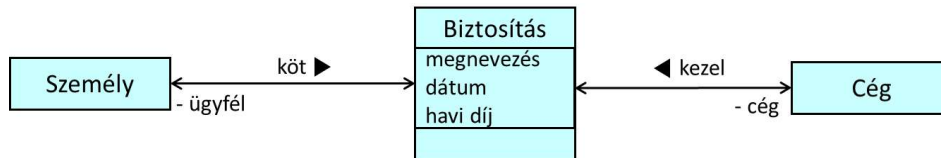
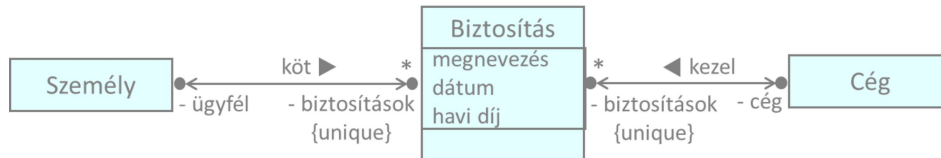
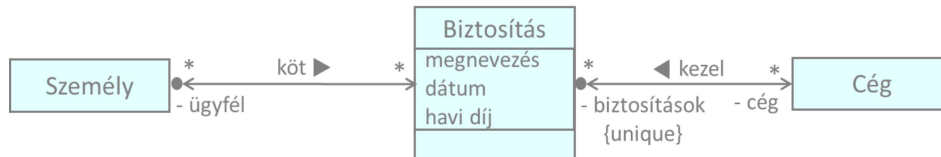
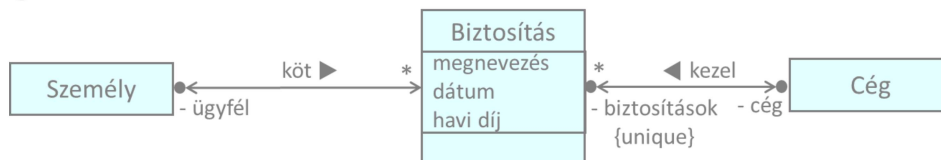
## 3. kérdés

0 / 1 pont

A személyek biztosításokkal foglalkozó cégekkel kötnek biztosításokat.



Melyik alábbi osztály diagram felel meg leginkább a fenti modellnek?

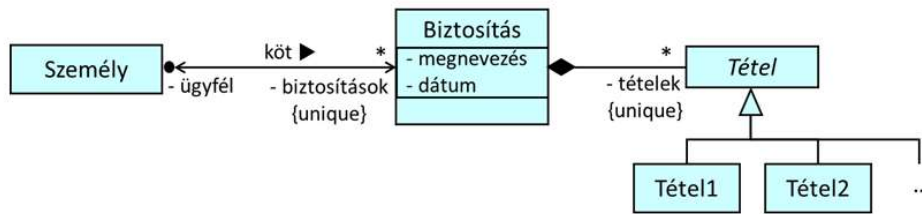
☒

☐

☐

☐


Részleges

## 4. kérdés

0.33 / 1 pont

Egy biztosítási szerződés keretében több különböző kártétel szerepelhet, amelyeket a biztosítás megkötésekor kell megadni.



Milyen metódus(ok) szükséges(ek) egy új biztosítás megkötéséhez? Az alábbi megoldások közül melyek helyesek?

```
Biztosítás osztály konstruktora:
Biztosítás(n:string, d:Dátum, ü:Személy, lista:string*)
{
    megnevezés := n; dátum := d; ügyfél := ü
    foreach(t in lista) loop
        switch ( t )
            case "tétel1": tétel := new Tétel1()
            case "tétel2": tétel := new Tétel2()
            ...
        endswitch
        tételek.Add(tétel)
    endloop
}
```



Személy osztály Kötés() metódusa:

```
Kötés(n:string, d:Dátum, lista:string*)
{
    new Biztosítás(n, d, this, lista)
}
```

Biztosítás osztály konstruktora:

```
Biztosítás(n:string, d:Dátum, ü:Személy, lista:string*)
{
    megnevezés := n; dátum := d; ügyfél := ü
    foreach(t in lista) loop
        switch ( t )
            case "tétel1": tétel := new Tétel1()
            case "tétel2": tétel := new Tétel2()
            ...
        endswitch
        tételek.Add(tétel)
    endloop
}
```



Személy osztály Kötés() metódusa:

```
Kötés(n:string, d:Dátum, lista:string*)
{
    b = new Biztosítás()
    b.megnevezés := n; dátum := d; ügyfél := ü
    foreach(t in lista) loop
        switch ( t )
            case "tétel1": tétel := new Tétel1()
            case "tétel2": tétel := new Tétel2()
            ...
        endswitch
        b.tételek.Add(tétel)
    endloop
}
```



```

Biztosítás osztály osztályszintű Kötés() metódusa:
Kötés(n:string, d:Dátum, ü:Személy, lista:string*)
{
    b = new Biztosítás()
    b.megnevezés := n; dátum := d; ügyfél := ü
    foreach(t in lista) loop
        switch ( t )
            case "tétel1": tétel := new Tétel1()
            case "tétel2": tétel := new Tétel2()
            ...
        endswitch
        b.tételek.Add(tétel)
    endloop
}

```

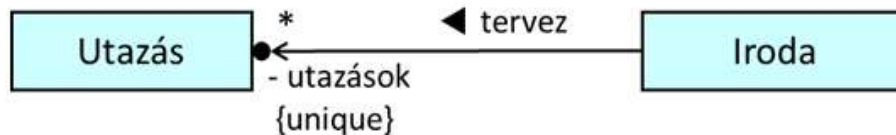


Helytelen

## 5. kérdés

0 / 1 pont

Egy utazási iroda csoportos körutazásokat hirdet. Milyen metódussal hozzunk létre új utazást?



Az *Utazás* osztályának egy metódusával, amelyik az `utazások.Add(new Utazás(...))` utasítást is tartalmazza.



Az *Iroda* osztály konstruktorával, amelyik az `utazások.Add(new Utazás(...))` utasítást is tartalmazza.



Az *Utazás* osztály konstruktorával, amelyik az `utazások.Add(new Utazás(...))` utasítást is tartalmazza.



Az *Iroda* osztályának egy metódusával, amelyik az `utazások.Add(new Utazás(...))` utasítást is tartalmazza.

## Helytelen

## 6. kérdés

0 / 1 pont

Egy utazási iroda csoportos körutazásokat hirdet, ahol egy körutazásra megadott létszámkorlátig jelentkezhetnek az utasok. Mi kerüljön a Jelentkezik() metódus törzsébe?



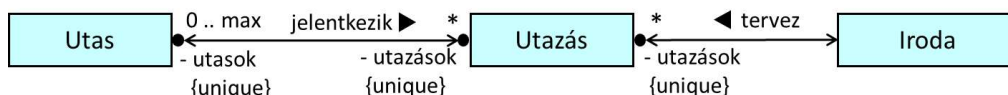
- ☒ if p in utas then error endif; utas.Add(p)
- ☐ utas.Add(p)
- ☐ if (|utas|=max) or (p in utas) then error endif; utas.Add(p)
- ☐ if |utas|=max then error endif ; utas.Add(p)

## Helytelen

## 7. kérdés

0 / 1 pont

Egy utazási iroda csoportos körutazásokat hirdet, ahol egy körutazásra megadott létszámkorlátig jelentkezhetnek az utasok. Hogyan válaszoljuk meg azt a kérdést, hogy hány olyan utazás van, ahol van még szabad hely?



Ehhez a *Utas* osztály alábbi törzsű publikus metódusára van szükség:

☐ **return**  $\sum_{\substack{ut \in utazasok \\ |ut.utasok| < max}} 1$

☒ Ehhez az *Utazás* osztály alábbi törzsű publikus metódusára van szükség:

**return**  $\sum_{\substack{ut \in utazasok \\ |ut.utasok| < max}} 1$

☐ Ehhez az *Iroda* osztály alábbi törzsű publikus metódusára van szükség:

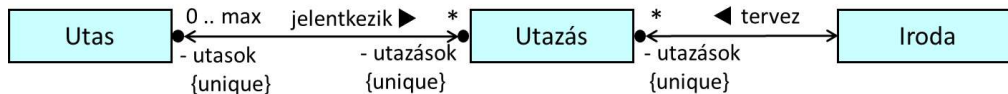
**return**  $\sum_{\substack{ut \in utazasok \\ |ut.utasok| < max}} 1$

☐ Ezt a kérdést nem lehet megválaszolni a modell alapján.

## 8. kérdés

1 / 1 pont

Egy utazási iroda csoportos körutazásokat hirdet, ahol egy körutazásra megadott létszámkorlátig jelentkezhetnek az utasok.



Hogyan válaszoljuk meg azt a kérdést, hogy melyik utazásra jelentkezett a legtöbb utas?



Ehhez az *Iroda* osztály alábbi törzsű publikus metódusára van szükség:

```
return MAXut ∈ utazások |ut.utasok|
```

Ehhez az *Iroda* osztály alábbi törzsű publikus metódusára van szükség:

```
if |utazások| = 0 then error end
return MAXut ∈ utazások ut.Hany()
```

és az *Utazás* osztály *Hany()* metódusának törzse: **return** |utasok|

Ehhez az *Iroda* osztály alábbi törzsű publikus metódusára van szükség:

```
if |utazások| = 0 then error end
return MAXut ∈ utazások |ut.utasok|
```

Ehhez az *Iroda* osztály alábbi törzsű publikus metódusára van szükség:

```
return MAXut ∈ utazások ut.Hany()
```

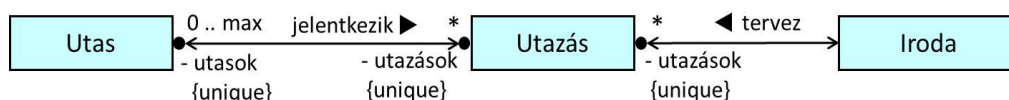
és az *Utazás* osztály *Hany()* metódusának törzse: **return** |utasok|

Helytelen

## 9. kérdés

0 / 1 pont

Egy utazási iroda csoportos körutazásokat hirdet, ahol egy körutazásra megadott létszámkorlátig jelentkezhetnek az utasok.



Hogyan válaszoljuk meg azt a kérdést, hogy igaz-e, hogy minden utazásra jelentkezett már olyan utas, aki több utazásra is regisztrált?



Ehhez az *Iroda* osztály alábbi törzsű publikus metódusára van szükség:

**return**  $\forall \text{SEARCH}_{ut \text{ in utazasok } ut.Van()}$   
 ahol az *Utazás* osztály *Van()* publikus metódusa: **return**  
 $\sum_{e \text{ in utasok}} 1$   
 $|e.utazasok| > 1$



Ehhez az *Iroda* osztály alábbi törzsű publikus metódusára van szükség:

**return**  $\forall \text{SEARCH}_{ut \text{ in utazasok } (\text{SEARCH}_{e \text{ in ut.utasok } e.UtakDb() > 1)}$   
 ahol az *Utas* osztály *UtakDb()* publikus metódusa: **return**  $|ut \text{ azások}|$



Ehhez az *Iroda* osztály alábbi törzsű publikus metódusára van szükség:

**return**  $\forall \text{SEARCH}_{ut \text{ in utazasok } (\text{SEARCH}_{e \text{ in ut.utasok } |e.utazasok| > 1)}$



Ehhez az *Iroda* osztály alábbi törzsű publikus metódusára van szükség:

**return**  $\forall \text{SEARCH}_{ut \text{ in utazasok } ut.Van()}$   
 ahol az *Utazás* osztály *Van()* publikus metódusa: **return**  
 $\sum_{e \text{ in utasok}} 1 > 0$   
 $e.UtakDb() > 1$   
 és az *Utas* osztály *UtakDb()* publikus metódusa: **return**  $|utazások|$

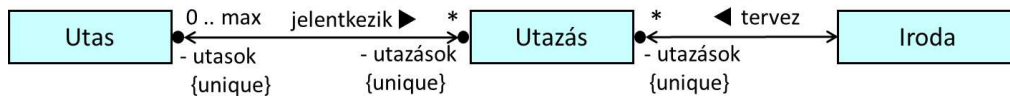
Helytelen

10. kérdés

0 / 1 pont

Egy utazási iroda csoportos körutazásokat hirdet, ahol egy körutazásra megadott létszámkorlátig jelentkezhetnek az utasok. Keressük meg azt az

utast, aki a legtöbb utazásra jelentkezett?


☐

```

max := 0
foreach út in utazások loop
  db, utas := MAXu in út.utasok | u.utazások
  if db>max then max, maxutas := db, utas endif
endloop
if max=0 then error endif
return maxutas

```

☒

```

max := 0
foreach út in utazások loop
  db, utas := MAXu in út.utasok u.Utazások()
  if db>max then max, maxutas := db, utas endif
endloop
if max=0 then error endif
return maxutas

```

ahol az *Utas* osztály *Utazások()* publikus metódusa egy utas utazásainak sorozatát adja vissza.

☐

```

max := 0
foreach út in utazások loop
  db, utas := MAXu in út.Utasok() u.Utazások()
  if db>max then max, maxutas := db, utas endif
endloop
if max=0 then error endif
return maxutas

```

ahol az *Utazás* osztály *Utasok()* publikus metódusa egy utazás utasainak sorozatát,  
az *Utas* osztály *Utazások()* publikus metódusa egy utas utazásainak sorozatát adja vissza.

```
max := 0
foreach út in utazások loop
    foreach u in út.Utasok() loop
        if |e.Utazások()|>max then max, maxutas := |u.Utazások()|, u
    endif
endloop
if max=0 then error endif
return maxutas
```

ahol az *Utazás* osztály *Utasok()* publikus metódusa egy utazás utasainak sorozatát,  
az *Utas* osztály *Utazások()* publikus metódusa egy utas utazásainak sorozatát adja vissza.

Kvízeredmény: **2.33** az összesen elérhető 10 pontból