zh-elmélet-1

Határidő dec 20, 19:00 **Pont** 50 **Kérdések** 5
Elérhető dec 20, 16:00 - dec 20, 19:00 körülbelül 3 óra **Időkorlát** 60 perc

Instrukciók

Válaszolj minden kérdésre legalább 1000-2000 karakter terjedelemben! A vizsga ezen részéhez semmilyen (írott, nyomtatott, digitális, illetve hálózaton elérhető) segédanyagot nem lehet használni, saját kútfőből kell a válaszokat megfogalmazni.

Elégséges szintű 20 ponttól,

közepes szintű 25 ponttól,

jó szintű 33 ponttól,

jeles szintű 40 ponttól.

Ezt a kvízt ekkor zárolták: dec 20, 19:00.

Próbálkozások naplója

	Próbálkozás	Idő	Eredmény
LEGUTOLSÓ	1. próbálkozás	52 perc	0 az összesen elérhető 50 pontból *

* Néhány kérdés még nem lett értékelve

Ezen kvíz eredménye: **0** az összesen elérhető 50 pontból *
Beadva ekkor: dec 20, 16:52
Ez a próbálkozás ennyi időt vett igénybe: 52 perc

1. kérdés — Még nincs értékelve / 10 pont

Mik az altípusosság szabályai függvénytípusokra? Írd le szövegesen és formálisan is.

A beküldött megoldás:

A függvények értelmezési tartománya bővíthető, az értékkészlete szűkíthető származtatásnál. Ahogy előfeltételeik enyhíthetők, míg utófeltéelei

szigoríthatóak.

Formális paramétereiket (LSP szerint helyesen) kontravariánsan (specializáció során általánosabb lesz)(azonban ezt nem szokták támogatni a programozási nyelvek) bővíthetjük, míg visszatérési értékét kovariánsan szűkíthetjük. (kovariáns: együtt változó)

Általános képlet:

Ha A <: A' és B' <: B, akkor A' -> B' <: A -> B

Kovariancia:

Legyen A <: B, és legyen T -ben egy f függvény B visszatérési típussal. Az S <: T akkor is fennáll, ha S újradeklarálja f-et A visszatérési típussal.

pl.:

class T

feature f: B

class S inherit T redefine f end

feature f: A

Kontravariancia:

Legyen A <: B, és legyen T -ben egy r rutin A típusú paraméterrel. Az S <: T akkor is fennáll, ha S újradeklarálja r-et B paramétertípussal.

pl.:

class T

feature r(A)

class S inherit T redefine r end

feature r (B)

2. kérdés

Még nincs értékelve / 10 pont

Hogyan szabályozza az Eiffel a láthatóságot? Miben különbözik ez a más nyelvekben megszokottól? (Itt sok mindenről lehet beszélni! Ez talán a legnagyobb terjedelmű választ igénylő kérdés ebben a feladatsorban!)

A beküldött megoldás:

Eiffel-ben a láthatóságok csak feature-ökön lehet megadni és csak objektumokra értelmezhetjük (osztályokkal ellenben más programozási nyelvekben). Ezt szelektív láthatóságnak nevezzük.

3 fő feature láthatóságot különböztethetünk meg:

- **feature** (vagy) **feature {ANY}** - ez azt jelenti, hogy csak azok a típusú objektumok és leszármazottaik érhetik el / láthatják ezt a feature-t, melyek az ANY altípusai (szóval az összes)
  - Ez más objektumorientált nyelvekben (Pl. Java) a Publikus láthatóságnak feleltethető meg
- **feature {NONE}** (vagy) **feture {}** (régiesen) - ez azt jelenti, hogy az adott feature csak az adott osztályon belül lesz látható
  - Ez más objektumorientált nyelvekben (Pl. Java) a Privát láthatóságnak feleltethető meg
  -
- **feture {A}** - ez azt jelenti, hogy csak az A altípusai számára lesz látható az adott feature (vagy maga A és leszármazottai)
  - Ez más objektumorientált nyelvekben (Pl. Java) nem igazán feleltethető meg egy konkrét láthatóságnak, de pl. a Protected láthatóságot így lehet kifejezni vele: Ha mondjuk egy class A -ban deklaráljuk mint feature {A} akkor úgy fog viselkedni mint a protected láthatóság más nyelvekben

Így más, legtöbb objektumorientált nyelvekhez képest sokkal részletesebben adhatjuk meg egy feature láthatóságát az osztályinkban.

3. kérdés

Még nincs értékelve / 10 pont

Mik a ciklus konstrukció szerződés-összetevői? Mikor felel meg a ciklus a szerződésének?

A beküldött megoldás:

Eiffel-ben a ciklusok az alábbi felépítésűek:

- **from** {INIT}
  - összetet utasítások, kezdő értékek
- **invariant** {INV}
  - a ciklus invariánsa, BOOLEAN-ra kiértékelődnek
- **variant** {VAR}
  - a ciklus variánsa, egy érték, aminek folyamatosan csökkenie kell
- **until** {COND}
  - BOOLEAN utasítások, megállási feltétel
- **loop** {BODY}
  - összetett utasítások, a ciklus teste
- **end**

A ciklusok szerződés-összetevői az invariánsai (invariant) és a variáns állítása (variant).

A ciklus invariánsa az a feltétel amelynek a ciklus inicializálása után (from, INIT) és a ciklusmag végrehajtása előtt és után fenn kell állnia.

A ciklus variáns függvénye az az állítás, melynek folyamatosan csökkennie kell, hogy biztosítsuk a ciklus haladását.

Hoare hármasokkal invariáns:

{TRUE} INIT {INV}

{INV} BODY {INV}

Lényegében a ciklus szerződése megfelel Eiffel-ben az általános ciklus szerződéssel, csak nincsenek elő és utófeltételek:

1. {TRUE} INIT {INV}
2. -
3. {INV and not COND} BODY {INV}
4. INIT => VAR >= 0
5. forall v : {INV and not COND and VAR = v} BODY {VAR <= v - 1}

4. kérdés

Még nincs értékelve / 10 pont

Mik az expandált típusok, miben rejlik a specialitásuk? (Ábrázolás, értékadás, paraméterátadás, egyenlőségvizsgálat, másolás, öröklődés/altípusosság.)

A beküldött megoldás:

Eiffel-ben az expandált típusok és a közönséges típusok között az a fő különbség, hogy az expandált objektumok a stack-en jönnek létre, míg a NEM expandáltak a heap-en.

Osztályokra lehet megadni az 'expanded' kulcsszóval, hogy az adott objektum expandált legyen-e, vagy sem:

expanded class INTEGER

Expanded típusokat csak érték szerint lehet átadni (paraméter típusától függ), aliasing nem alkalmazható rájuk. Egyenlőséget is csak érték szerint, rekurzívan vizsgálhatunk rajtuk (ugye mivel referenciával nem rendelkeznek), másolásuk érték szerint, rekurzívan történik bármilyen esetben. Csak attached (nem Void) értékek lehetnek (nem üres, mivel nincs referencia). Örökölni belőlük csak privátan lehetséges, vagyis nem fog létrejönni típus - altípus reláció (öröklési hierarchia).

Egy kivétel: ha egy NEM expanded osztály tartalmaz egy expanded attribútumot, akkor az expanded osztály is a heap-en fog létrejönni, de minden más egyébként igaz lesz rá (ami a többi expanded típusra is).

5. kérdés

Még nincs értékelve / 10 pont

Mit értünk sekély-, illetve mély másoláson?

A beküldött megoldás:

Eiffel-ben sok 3 fő másolási lehetőségünk van objektumokra:

- **custom** - egyedileg megadható, felüldefiniálható
- **standard** - ez lesz a sekély (shallow) másolás
- **deep** - ez lesz a mély másolás

utasítások:

- is_equal(A)
- standard_is_equal(A)
- is_deep_equal(A)

kifejezések

- equal(A, B)

- standard_equal(A, B)
- deep_equal(A, B)

Sekély másolásnál NEM expanded osztályokon úgy történik, hogy végig iterálunk az osztály összes attribútumán és azt referencia szerint (= operátorral) összehasonlítjuk. Expanded osztálynál pedig érték szerinti, rekurzív összehasonlítás fog történni (~ operátor).

Mély másolásnál mind expanded és nem expanded típusokon úgy történik, hogy rekurzívan bejárjuk az összes attribútumát és azokat érték szerint összehasonlítjuk (~ operátor).

Kvízeredmény: **0** az összesen elérhető 50 pontból