zh-elmélet-2

Határidő jan 20, 15:00 **Pont** 50 **Kérdések** 5
Elérhető jan 20, 13:00 - jan 20, 15:00 körülbelül 2 óra **Időkorlát** Nincs

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.

Elégségest 20 ponttól,

közepest 25 ponttól,

jót 33 ponttól,

jelest 40 ponttól adok.

Ezt a kvízt ekkor zárolták: jan 20, 15:00.

Próbálkozások naplója

	Próbálkozás	Idő	Eredmény
LEGUTOLSÓ	1. próbálkozás	67 perc	34 az összesen elérhető 50 pontból

Ezen kvíz eredménye: **34** az összesen elérhető 50 pontból
Beadva ekkor: jan 20, 14:20
Ez a próbálkozás ennyi időt vett igénybe: 67 perc

1. kérdés

6 / 10 pont

Mit jelent a *bounded parametric polymorphism*? Milyen megszorításokat lehet tenni az Eiffelben?

A beküldött megoldás:

Eiffelben a bounded parametric polymorphism alatt olyan polimorfizmust értünk melyben az osztályok típusparaméterére teszünk megszorításokat. Ezzel biztosíthatjuk azt, hogy a kapott típusok már rendelkeznek(örököltek) a kívánt osztályoktól azaz rendelkeznek a kívánt feaureökkel melyekre az osztályunknak szüksége van, hogy dolgozni tudjon velük.

Egy egyszerű példával lehet ezt szemléltetni:

Amennyiben van egy SORT[T] osztályt implementálunk akkor érdemes megkötni, hogy a T típusparaméterben a T-re értelmezve legyen a rendezés művelete. Ezt így tehetjük meg SORT[T->COMPARABLE] egyszerre több megkötést is tehetünk, ilyenkor az {}-ek közé tesszük a megkötéseket ilyenkor ezek össze és-elődnek

2. kérdés

10 / 10 pont

Mire való az `old` kulcsszó az Eiffelben? Magyarázd el, fogalmazd meg a rá vonatkozó szabályokat, illusztráld használatát példán!

A beküldött megoldás:

Az old kulcsszóval a változás illetve nem változás tényét tudjuk kifejezni rutinokban. Olyan ez mint amikor rutin meghívása előtti pillanatban lévő értékekre hivatkozunk. Az old kulcsszót a szerződés utófeltételében (ensure) tudjuk felhasználni. Ekkor ha például az órán is látott egyszerű BANK_ACCOUNT osztály példájával élünk ahol van egy deposit eljárásunk. A deposit eljárásnak paraméterül adható a befizetni kívánt összeg miután lefutott az eljárás az ensure szerződésében megnézhetjük, hogy valóban annyi pénz adódott hozzá a számlánkhoz mint vártuk.

deposit(amount:INTEGER)

require

positive_amount: amount>0

do

balance := balance + amount

ensure

deposit_valid : balance = old balance + amount

end

A fenti példával most a változás tényét mutattuk meg.

A azokat a mezőket melyeket a rutin végrehajtása nem értint(nem változtat meg) frame condition-nek hívjuk erre is alkalmas az old kulcsszó a fenti példával élve tegyük fel, hogy a deposit eljárás csak a balance értékét változtatta. Ekkor ezt a következőképpen adjuk meg a szerzősédbe:

frame: strip(balance) ~ old (strip (balance))

A strip-el leválasztottuk a balance mezőt és a ~-vel pedig azt mondjuk, hogy minden más egyenlő maradt a régi értékével

Az ECMA szabvány megengedi az only keyword használatát is. az only balance eqvivalens a strip(balance) ~ old (strip (balance)) -el azonban ezt az eiffel studio még nem támogatja

3. kérdés

7 / 10 pont

Mi a különbség az `=`, a `~`, az `is_equal`, az `equals`, a `standard_is_equal` és az `is_deep_equal` között? Mi közük van egymáshoz?

A beküldött megoldás:

=: Alapértelmezetten referenciák egyenlőségét nézi, két objektum akkor egyenlő ha megegyezik a referenciájuk.

~: Tartalmi egyezést vizsgál két objektum akkor egyenlő ha a benne lévő mezők referencia szerint (sekély vizsgálat) megegyeznek. Tehát itt például lehet, hogy a sima = hamisra értékelődik ki mert a két objektum referenciája nem egyenlő de attól még a tartamuk megegyezhet.

is_equal : ez egy deferred feature. A programozó itt kapja meg a lehetőséget, hogy saját összehasonlítást adjon meg. Ez a függvény az equals meghívása során meghívásra kerül

equals: Ez egy frozen feature. Először megvizsgálja, hogy a kapott objektum Void-e mert akkor csak akkor lehetnek egyenlőek ha mind a kettő void amennyiben nem akkor ezek után referencia szerint összehasonlítja őket ha ez sem teljesül akkor meghívja az is_equal függvényt melyet mint előbb említettem mi adhatunk meg.

is_deep_equal : Ez a frozen függvény a mély tartalmi egyenlőségvizsgálatot adja meg. Mély egyenlőségvizsgálat alatt azt értjük, hogy rekurzívan nézzük meg az objektum mezőin aztán annak a mezőin, hogy egyenlőek-e.

standard_is_equal: Ez az alapértelmezett tartalmi egyenőlés azaz az objektum mezőinek sekély, referencia szerinti összehasonlítása

4. kérdés

6 / 10 pont

Mi a `default_create`? Mire való, miért jó? Járd körül a kérdést alaposan!

A beküldött megoldás:

A default create minden osztály alapértelmezett creation procedureje melyet az ANY osztálytól öröklünk meg, ha az osztályban nem adunk meg egy saját creation proceduret sem. Erre azért van szükség, mert az Eiffel előírja számunkra, hogy minden osztálynak kell rendelkeznie legalább egy creation procedure-el. Amikor például a kódunk-ban a "create valami" utasítást adjuk ki akkor mivel nem specifikáltuk, hogy melyik creation proceduret szeretnénk meghívni ezért automatikusan a default hívódik meg. Emellet a default_create-et át is nevezhetjük, figyelni kell, hogy ettől még a default create hívódik meg (nem a nevétől az ami). Illetve redefine-t is alkalmazhatunk rá mivel a default_create alap esetben mindent az alapértemezett értékével tölt fel például INTEGER-t 0 val egyéb objektumokat Void-al és ez nem feltétlenül előnyös számunkra.

5. kérdés

5 / 10 pont

Mit jelent a *polymorphic CAT-call*?

A beküldött megoldás:

A polymorphic CAT-call -ban a CAT jelentése Changed Availability of Type. Ez alatt azt értjük, hogy az öröklödés során a megörökölt mezőnek csökken a láthatósága vagy ha a paraméterül kapott típusok szűkülnek. Ez amellett, hogy ellent mond az LSP-nek megtermti a lehetőségét annak, hogy kicselettük a statikus típusellenőrzést. Könnyen látható, hogy ha például egy olyan típust adunk meg paraméterül a szűkült paraméternek mely nem definiálja az adott feature-t akkor futás közben ha ezt meghívjuk akkor nagyon valószínű, hogy futási idejű hibát kapunk. Az Eiffel erre jelenleg nem kínál megfelelő megoldást, sőt igazából egy nyelv sem... Annak ellenére, hogy elméletileg lehetséges egy jól működő CAT-call ellenőrző elkészítése, ez számításilag nagyon költséges lenne.

Kvízeredmény: **34** az összesen elérhető 50 pontból