Objektum orientált fejlesztés	**Vizsga 2005. jan. 4.**	Jegy:
Név:	Neptun:	Pont:

1. Mit eredményez az alábbi programrészlet lefutása ? Mi lesz a következménye ? Hogyan tehető a program korrektté ?(6 pont)

```
Object object = db.getRoot("username");
ObjectStore.destroy(object);
```

Töröltük a gyökérként referált objektumot.
getRoot nem létező objektumra fog mutatni. Exception lesz, ha rajta keresztül akarunk elérni valamit.

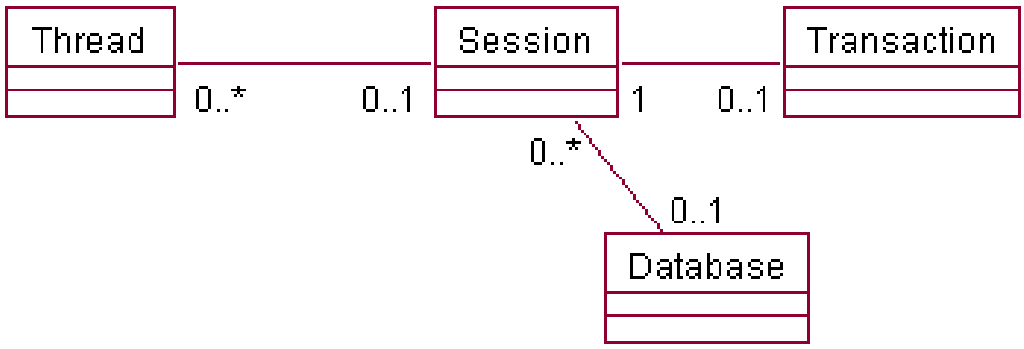
Gyökérnek új értéket adunk, pl:

```
db.setRoot("username", null); vagy
db.setRoot("username", Ujgyökérnekvaló);
```

Mindegyik válasz 2 pont

---

2. Rajzoljon UML osztálydiagramot, amely leírja az ObjectStore-ban értelmezett szál, session, tranzakció és adatbázis közötti kapcsolatokat ! (3 pont)



Mindegyik korrekt reláció 1 pont

---

3. A CORBA bevezette a ”pszeudo object” fogalmát. Miben különbözik ez a ”hagyományos” objektumtól (2 pont) Nevezzen meg legalább két pszeudo objektumot ! (2 pont)

operations are implemented in libraries – nincs class file
some constraints: inheritance, instancing

példa: CORBA.Object, ORB, POA

minden helyes válasz 1 pont

4. Adott az alábbi IDL, ahol az `AOp2` metódussal beállítható értéket lehet az `AOp1`-gyel lekérdezni, valamint a `BOp` metódussal az aktuális értéket lehet megnövelni a paraméter értékével.

```
module zh {
    interface A {
        long AOp1();
        void AOp2(in long a);
    };
    interface B: A {
        void BOp(in long b);
    };
};
```

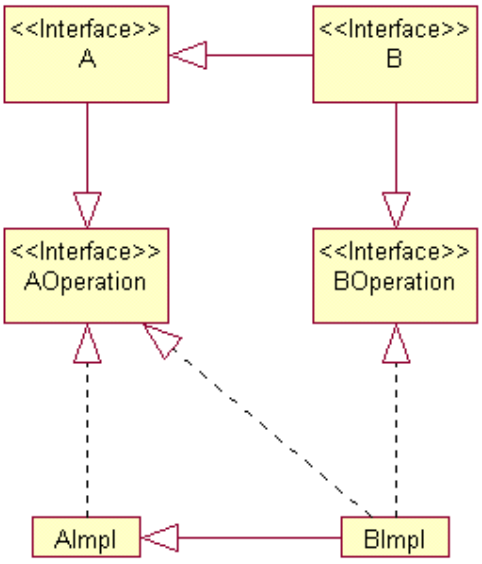
Implementálja az `A` és `B` interfészeknek megfelelő objektumokat (`AImpl` és `BImpl`, ez utóbbi örökölje `A` metódusait) ! (2 pont)

Rajzolja fel azt az osztálydiagramot, amelyen `A`, `B`, `AImpl`, `BImpl` valamint ezeknek valamennyi alkalmazásfüggő őse (osztály, interfész) szerepel ! (4 pont)

Legyen egy szerver program, amelyik létrehoz egy `B` típusú szervantot. Írja meg azt a programrészletet, amellyel létrehozza `B`-t és szervantként a POA-hoz kapcsolja ! (3 pont). Feltételezheti, hogy rendelkezik egy `orb` nevű `ORB` objektummal és az IDL compiler által előállított valamennyi osztállyal.

```
public class AImpl implements AOperations {
  protected int aa;
  public` int AOp1() { return aa; }
  public void AOp2(int a) { aa = a; }
}
```
(1 pont)

```
public class BImpl extends AImpl implements AOperations, BOperations {
  public void BOp(int a) { aa += a; }
}
```
(1 pont)



(4 pont)

```
  BImpl bImpl = new BImpl();
  BPOATie tie = new BPOATie(bImpl);
  B b = tie._this(orb);
```
(3 pont)

5. Mit nevezünk aspektusnak ? Adjon rá példát ! Hogyan kapcsoljuk az aspektust a programhoz ? (3 pont)

Aspektus = olyan szempont, amely több komponens adaptálását teszi szükségessé. (cross-cutting concern)
Például: debug, perzisztencia, konkurrencia kezelése
Az aspektusokat a programba szőjük (weaving)
minden helyes válasz 1 pont

6. Jellemezze röviden az *Event Service*-ben definiált két modellt és két szerepet! (6 pont)

push model: a termelő az eseményeket kérés nélkül a fogyasztónak adja, utobbi push() metódusának meghívásával. A termelő aktív, a fogyasztó passzív. (2 pont)
pull model: a fogyasztó, ha eseményt kíván kapni, meghívja a termelő pull() metódusát. A termelő passzív, a fogyasztó aktív. (2pont)

termelő: az a szereplő, aki az eseményeket előállítja. (1 pont)
fogyasztó: az a szereplő, aki az eseményeket megkapja. (1 pont)

7. Írja le röviden, hogy ha egy *Event Service*hez megírt alkalmazást minimális módosítással a *Notification Service*-zel szeretnénk használni, akkor a következő két sort mire kell cseréni (6 pont):

```
org.omg.CORBA.Object obj =
     orb.resolve_initial_references("EventService");
EventChannel ec = EventChannelHelper.narrow(obj);
```

```
org.omg.CORBA.Object obj =
     orb.resolve_initial_references("NotificationService");
EventChannelFactory ef =
     EventChannelFactoryHelper.narrow(obj);
EventChannel ec = ef.get_event_channel(id);
```

(soronként 2 pont)

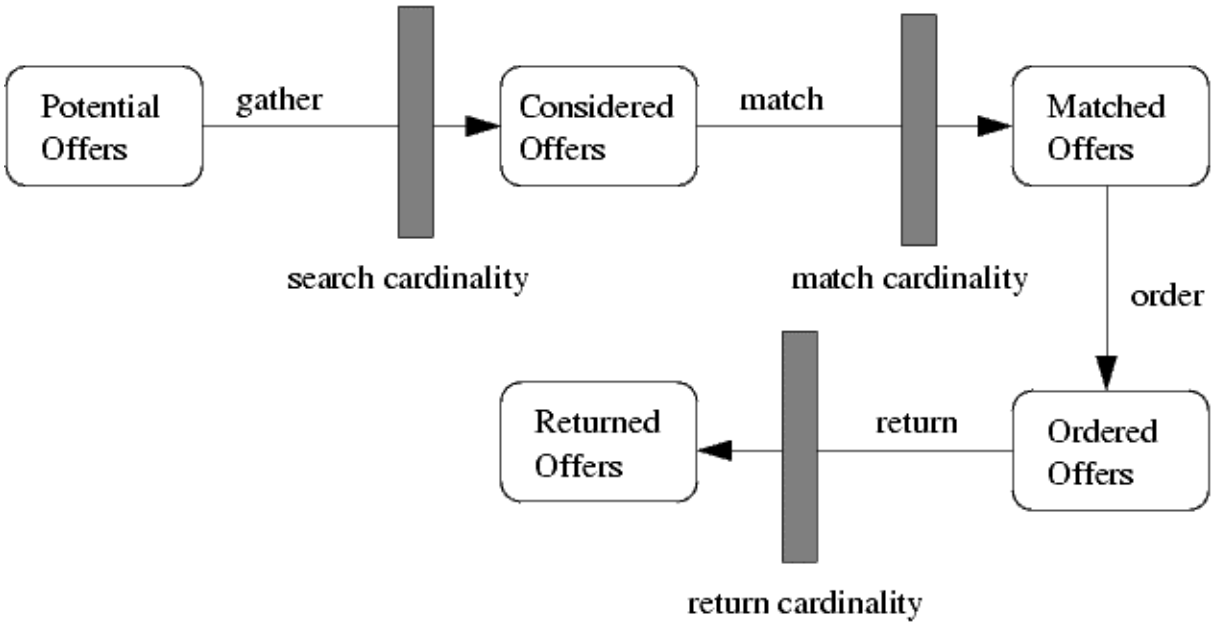
8. Mi történik, ha a Trader több ajánlatot talált, mint amennyi az általunk készített szekvencia mérete? (4 pont)

egy, a trader által visszaadott iterátor objektum segítségével a szekvenciát újratölthetjük.

(trader által visszaadott 2 pont, újratölt 2 pont)

9. A Trader Serviceben, a keresések során milyen megszorító számosságokat tudunk definiálni, és ezek az ajánlatok feldolgozása során hol kapnak szerepet (ábrán szemléltetve) (9 pont)

search, match és return cardinality



(minden számosság és jó doboz 1 pont, szűrők jó helyen +1)

Eredmények értékelése:

Pontszám	***Osztályzat***
21 -	***2***
28 -	3
35 -	4
42 -	5