UML osztály diagram – Feladatok

2008.01.22 – 8. Feladat

Rajzoljon UML 2 osztálydiagramot! A metódusokat és attribútumokat nem kell jelölje! Csak a vastagon szedett classifier-ek szerepeljenek a diagramon! (8 pont)

Egy cégnél nyilvántartják az **alkalmazottak** és a **projektek** adatait. Az alkalmazottak projektekhez vannak rendelve, minden projektnél különböző beosztásban. Minden projekten dolgozik legalább egy alkalmazott, de lehet olyan alkalmazott, aki éppen nincs projekthez rendelve. A **beosztás** határozza meg például, hogy mennyi bónuszt kap az alkalmazott az adott projekt sikere esetén. Mind a projekt, mind a beosztás jogilag ellenőrzendő (megvalósítják az **ellenőrzendő** interfészt). A **jogász** (aki persze a cég alkalmazottja is egyben) dolga, hogy az ellenőrzéseket elvégezze. A **registryben** tárolják az alkalmazottak azonosítóinak és az alkalmazotti adatoknak az összerendelését. A registry az egyszerűség kedvéért a **HashTable** (K kulcs és X érték paraméterő) template osztályt példányosítja, ahol a kulcs az azonosító, az érték az alkalmazott adata.

2008.06.10 – 9. Feladat

Egy dokumentumkezelő rendszerben heterogén kollekcióként névvel ellátott dokumentumokat tárolunk. A dokumentum lehet irat vagy mappa. A mappa további dokumentumokat tárolhat, de maximum 10-et. A rendszernek képesnek kell lennie arra, hogy később további dokumentumfajtákkal (fotó, hangszalag stb. bővítsük). A dokumentumokat ki tudjuk nyomtatni (**nyomtat**), az iratokat alá lehet írni (**aláír**), a mappákba újabb dokumentum helyezhető (**hozzáad**). A rendszer nyilvántartja a dokumentum-módosításokat is. Minden dokumentum a módosításról értesíti a rendszert (**változás** metódus), és átadja a módosítás adatait leíró objektumot. Ebben a módosuló dokumentum megadja a nevét és a módosítás idejét, de ezen kívül a dokumentum típusától függő egyedi adatok is szerepelhetnek (irat esetén az aláíró neve, mappa esetén a mappa mérete).

2009.01.06 – 6. Feladat

Rajzoljon UML2 osztálydiagramot az alábbi Java programrészlet alapján! Az osztálydiagram ne tartalmazzon olyasmit, ami a kódból nem olvasható ki ! A metódust TILOS feltüntetnie!

```
class MyClass {

	public void mx(YourClass y) { }

}
```

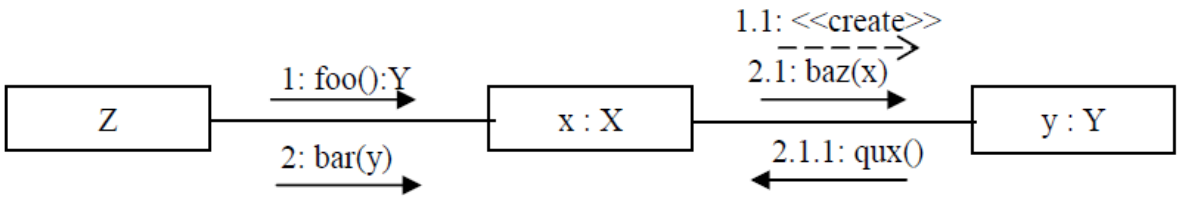
2010.05.26 – 7. Feladat

Rajzoljon UML2 osztálydiagramot az alábbi történet alapján! Jelölje a számosságokat is! (8 pont)

Az egyetemi polgárokat, akiknek nyilvános a neve és a neptunkódja, a Neptun rendszer tartja nyilván, mégpedig a polgáronként egyedi neptunkód alapján. Polgár az oktató és a hallgató is. Az oktatónak nyilvántartjuk a szobaszámát, a hallgatónak az email címét (privát adat). A Neptunban tároljuk a tárgyakat is egyedi neptunkóddal. A tárgyaknak szintén ismerjük a nevét és neptunkódját, valamint az értük kapható kreditek számát. Egy hallgató több tárgyat is felvehet (hallgatja), egy tárgyra több hallgató is járhat. Ezen kívül egy adott hallgató egy adott tárgyra kapott jegyét és a megszerzés évét is nyilvántartjuk. Egy tárgyat legalább egy oktató oktat, egy oktatónak pedig lehet több tárgya is (maximum 5), de van akinek egy sincs. A Neptunba fel tudunk venni új tárgyat és polgárt.

2010.06.01 – 6. Feladat

Adott a következő UML2 kommunikációs diagram.



Feltételezve, hogy a fenti diagramon szereplő objektumok osztályainak nincsenek – a diagramból nem kiolvasható – további metódusai, közöttük nincs más egyéb kapcsolat (pl. öröklés), az alábbi ábrát korrekt UML2 osztálydiagrammá alakítva ábrázolja az osztályokat a metódusok szignatúráival együtt, valamint a két osztály közötti kapcsolatot!

Mi a kapcsolat az *operáció* és *metódus* között?

2010.12.21 – 8. Feladat

Készítsen UML 2 osztálydiagramot (class diagram) az alábbi leírás alapján! Használja a kövéren szedett kifejezéseket! Ahol lehet, adja meg a paraméterek, attribútumok, stb típusát is! (8 pont)

Az OOniX operációs rendszerben **folyamatok**, **fájlok** és hálózati kapcsolatok (**socket**) vannak. A folyamatoknak van azonosítója (**pid**), tulajdonosa (**uid**). Egy folyamatból a **fork()** metódussal lehet újat létrehozni. Minden folyamat ismeri a közvetlen **ősét** és a **gyerekeit**. A fájloknak van egy senki más által nem látható **inode** száma, és lekérdezhető a **méretük**. A fájlok többfélék lehetnek: **könyvtárak**, amelyek más fájlokat tartalmazhatnak (a **nevük** alapján), **reguláris** fájlok, amiknek van **típusa**, stb. Minden fájl egy könyvtár része. A folyamatok egyformán kezelhetnek socketet és fájlt is, de csak egy közös interfészt (**stream**) látnak belőlük, amiken bájtokat lehet **olvasni** és **írni**. Az ilyen objektumokról a folyamatnak van egy listája, aminek az elemeit fájlleíróval (**fd**) azonosítja. A folyamatok létrehozásakor egy (a folyamatok számára közös) számláló (**lastpid**) növekszik, ez lesz az újonnan létrehozott folyamat azonosítója. A folyamatoknak más folyamatok tudnak üzenni a **signal()** üzenet meghívásával (egy darab egész típusú paramétere van). A lastpid és a signal csak folyamatból (és esetleges leszármazottjából) látható.

2011.06.07 – 9. Feladat

Rajzoljon az alábbi programrészletnek megfelelő UML2 osztálydiagramot! Csak olyan jellemzőket ábrázoljon, amely a kódrészletből kiolvasható!

```
public interface Pelda {.....}
public class Hallgato {
	private Jegyzet j;
	public int vizsgaz(Pelda p) {
		int n = j.puskaz(p); ...}
}
```

2012.01.03 – 9. Feladat

Az alábbi Java kódrészletek alapján rajzoljon UML2 osztálydiagramot!

```
class Q implements S extends F {
	protected static int count;
	public K key;
	public int foo(X eks) {
		eks.bar();
	}
}

interface X extends S {
	void bar();
}

interface S {
	long round(double d);
}

class F {
	private long l;
	long baz() { return l; }
}

class K {
	F qux() { return new F(); }
}
```

2012.05.22 – 10. Feladat

Rajzoljon UML2 osztálydiagramot az alábbi Java kódrészlet alapján!

```
public interface W {
	public double foo();
}

public class X implements W {
	private java.util.Map<String, Y> hm;
	public double foo() {return 1; }
}

public class Z {}

public class Y extends Z {
	protected W parent;
	public void bar(W p) { parent = w; }
	static private int cnt;
}

public class F {
	rotected java.util.List<Z> tool;
	public X create() { return new X(); }
}
```

asd