Programozás III

SOK MINDEN

AMI KIMARADT

Sok minden

Néhány további figyelemfelhívás, vizsga-elvárás:

Tiszta kód elvek (nagyon érintőlegesen)

Robert C. Martin:Tiszta kód - Az agilis szoftverfejlesztés kézikönyve

http://www.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0103_21_programozasi_technologiak/ch02s04.html

https://www.refaktor.hu/tiszta-kod-1-resz-teszteljek-vagy-ne-teszteljek/

SOLID TERVEZÉSI ELVEK

SOLID:

- Single responsibility,
- Open-closed,
- Liskov substitution,
- Interface segregation and
- Dependency inversion

OOP tervezési „ökölszabályok”.
Célja a szoftvertisztítás. Az agilis szoftverfejlesztés része.

SINGLE RESPONSIBILITY PRINCIPLE

„Egyszeres” felelősség:

A felelősség: ok a változtatásra.

Ha egy osztály több dologért is felelős, akkor több ok lehet a változtatásra, sőt, több, egymástól független ok.

Alapelv: egy osztály csak egyetlen dologért legyen felelős, vagy ha több felelőssége van, akkor ezek nagyon szorosan összekapcsolódnak.

SINGLE RESPONSIBILITY PRINCIPLE

OPEN-CLOSED PRINCIPLE

Megvalósítás: nagyfokú absztrakció.

Interface, absztrakt osztály/metódus,

öröklődés, kompozíció.

OPEN-CLOSED PRINCIPLE

Az egyedek (osztály, csomag, metódus) legyenek nyitottak a bővítésre, de zártak a módosításra.

Nyitott a bővítésre: Könnyen hozzáadható új funkcionalitás.

Zárt a módosításra: Bővítéskor nem kell átírni a forráskódot.

OPEN-CLOSED PRINCIPLE

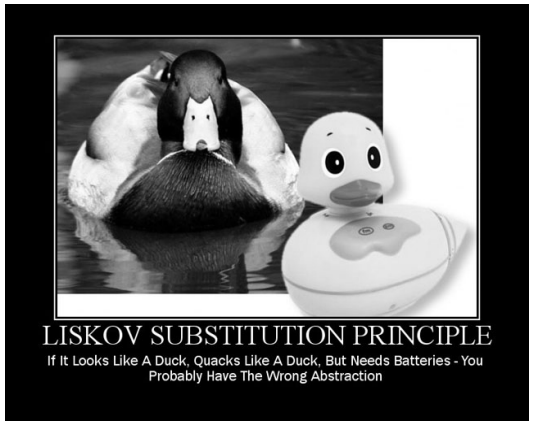
LISKOV SUBSTITUTION PRINCIPLE

Az ősosztály mindig helyettesíthető legyen az utód-osztállyal.

Ennek feltétele:

Ha a leszármazás során nem csökken az ős funkcionalitása és megfelel a logikus elvárásoknak.

LISKOV SUBSTITUTION PRINCIPLE



LISKOV SUBSTITUTION PRINCIPLE

Néhány javaslat:

„Tell, don't ask": Semmit se feltételezzünk az osztályok belső felépítéséről. Nem kell ismernünk az implementációt ahhoz, hogy kérni tudjuk tőlük a feladatuk elvégzését.

Helyes hierarchia: Ha pl. van két hasonlóan viselkedő osztály, de nem helyettesíthetőek egymással, akkor vezessünk be egy ős-osztályt, amelyből mindketten örökölhetnek.

INTERFACE SEGREGATION PRINCIPLE

Az alapelv: a kliensek ne függjenek olyan metódusoktól, amelyeket nem használnak.
Pl. egy interfész vagy absztrakt osztály nem minden metódusát implementáljuk, vagy csak töredékesen használjuk ki egy-egy osztály funkcionalitását.

Megoldás: Több kisebb interfész.

INTERFACE SEGREGATION PRINCIPLE

DEPENDENCY INVERSION PRINCIPLE

Pl: ehelyett:

```
public class Volvo {
    B20 engine;

    public Volvo() {
        engine = new B20();
    }
}
```

inkább ez:

```
public class Volvo {
    IEngine engine;

    public Volvo(IEngine engine) {
       if (engine == null) throw new NullPointerException("engine");
       this.engine = engine;
    }
}
```

mert ekkor pl. így is hívható:

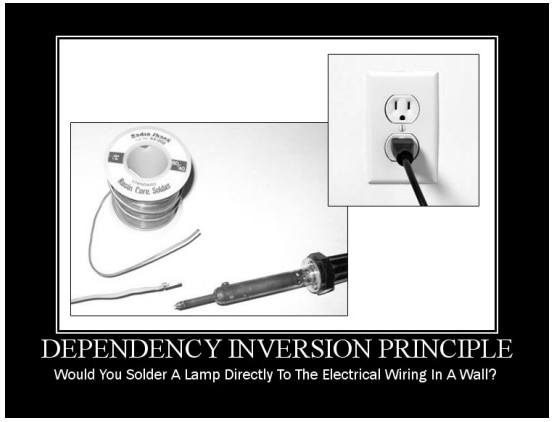
```
Volvo myVolvo = new Volvo(new BigBadV12());
```

DEPENDENCY INVERSION PRINCIPLE

Az elv lényege: a függőségek kialakítását a hívó félre bízzuk, és ne magára az osztályra.

Vagy másképp: a magas szintű modulok lehetőleg ne az alacsony szintűektől, hanem valamilyen absztrakciótól függjenek.

DEPENDENCY INVERSION PRINCIPLE



IRODALOM

http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf

http://www.hanselminutes.com/145/solid-principles-with-uncle-bob-robert-c-martin
http://s3.amazonaws.com/hanselminutes/hanselminutes_0145.pdf

http://zeroturnaround.com/rebellabs/object-oriented-design-principles-and-the-5-ways-of-creating-solid-applications/

http://blog.gauffin.org/2012/05/solid-principles-with-real-world-examples/

TDD

A TDD egy olyan szoftverfejlesztési módszer, ahol nagyon rövid fejlesztési ciklusok követik egymást.
Ezek általában a következők:
Tesztet írunk az új funkcióhoz.
Tesztek futtatása. Az újonnan írt teszt meghiúsul.
Funkció implementálása.
Tesztek újbóli futtatása. Tovább: ha minden teszt sikeres.
Refakorálás

Ezek a lépések ismétlődnek.

TDD (Test Driven Developement)

TDD = TFD + Refactoring

TFD: test first development

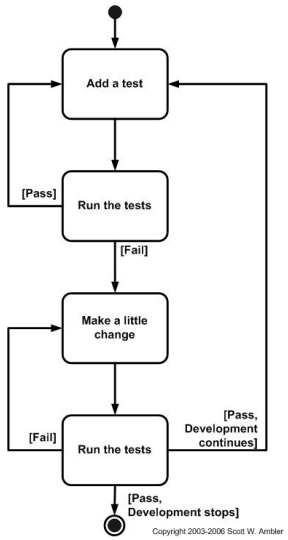
TDD: test driven development

A TDD alapötlete: fejlesztés közben először a (unit) teszteket írjuk meg, és csak ezután látunk neki a valós funkciót megvalósító kódnak.

TDD

Ez a módszer
- magas teszt lefedettséget eredményez,
- a fejlesztőt arra ösztönzi, hogy kis, minél kevesebb külső függőséggel rendelkező önálló részekre bontsa fel a programot, így azok könnyen tesztelhetőek.

TFD



KONKLÚZIÓ

If it's worth building, it's worth testing.
If it's not worth testing, why are you wasting your time working on it?

(http://www.agiledata.org/essays/tdd.html)

Példák:
http://ivarconr.wordpress.com/2012/01/26/tdd-by-example-factorial/
http://technologyconversations.com/2013/12/20/test-driven-development-tdd-example-walkthrough/

TDD

Kent Beck (az eXtreme Programming (XP) egyik népszerűsítője), két egyszerű TDD szabályt definiál:

1.Csak akkor írj új programkódot, ha a teszt sikertelen.

2.Eliminálj minden duplikátumot.

(http://www.agiledata.org/essays/tdd.html)

A TDD egy érdekes mellékhatása: 100%-os lefedettségi tesztet eredményez. A kód minden egyes sorát teszteljük.

AJÁNLOTT OLDALAK

https://www.iit.bme.hu/~softlab3/lab10/JUnitGuide.pdf

http://www.vogella.com/tutorials/JUnit/article.html

http://tutorials.jenkov.com/java-unit-testing/database-testing-crud.html

+ google

AJÁNLOTT OLDALAK

http://www.agiledata.org/essays/tdd.html

http://ivarconr.wordpress.com/2012/01/26/tdd-by-example-factorial/

http://butunclebob.com/ArticleS.UncleBob.TheBowlingGameKata

http://www.objectmentor.com/resources/articles/xpepisode.htm

http://technologyconversations.com/2013/12/20/test-driven-development-tdd-example-walkthrough/

http://osherove.com/tdd-kata-1/

Kent Beck: Test-Driven Development By Example

Steve Freeman, Nat Pryce: Growing Object-Oriented Software, Guided by Tests

+ google

AJÁNLOTT OLDALAK

http://rogerdudler.github.io/git-guide/

https://try.github.io/levels/1/challenges/1

http://ithub.hu/blog/post/Amit_tudnod_kell_fejlesztokent_IV_resz_Verziokezeles/

http://git-scm.com/

http://www.math.bme.hu/~balazs/git/gitcml.html

http://vili.pmmf.hu/portal/hu/web/zamek/home/-/document_library_display/WIw1/view/13601

http://www-cs-students.stanford.edu/~blynn/gitmagic/

http://www.codeproject.com/Articles/457305/Basic-Git-Command-Line-Reference-for-Windows-Users

http://msysgit.github.io/

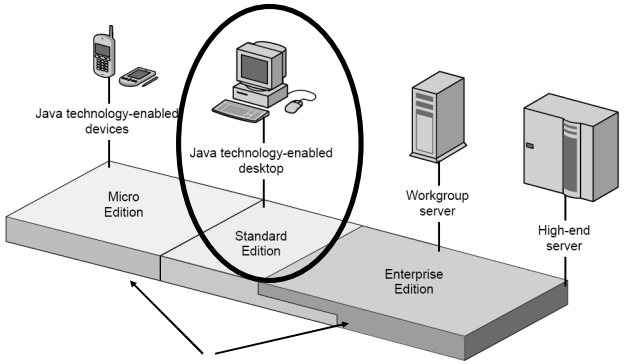
+ google

VIZSGAFELADAT - ELVÁRÁSOK

1. Jól működő, ötletes projekt ☺
2. Tesztelés
3. Dokumentálás
   - beszédes változónevek, konvenciók betartása
   - javadoc kommentek (csak ilyen kommentek)
   - rendes javadoc készítése
   - szöveges dokumentáció készítése
4. Pozitívum: verziókezelő használata

AMI KIMARADT

2. További, kevésbé gyorsan pótolható témák



kimaradt ☹

AMI KIMARADT

Java EE

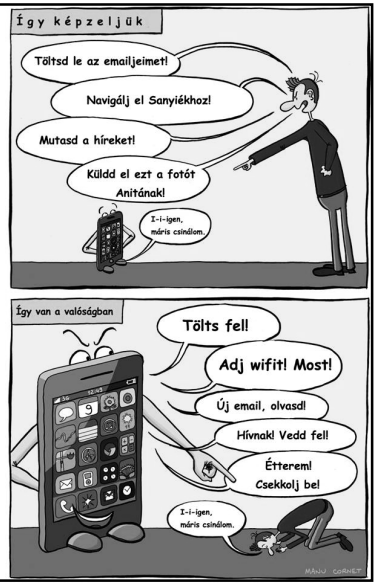
Sok munkával tanulható meg, de megéri.

JAVAslat: JAVA2 tárgy!

AMI KIMARADT

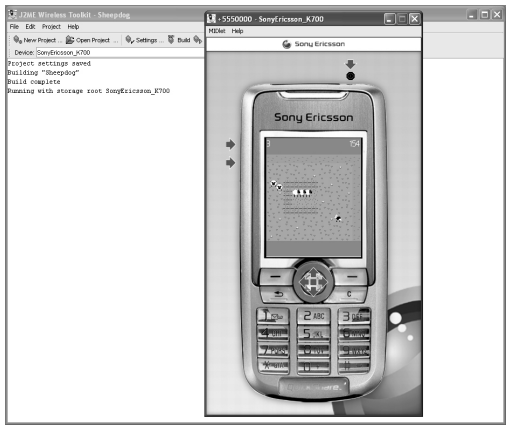
J2ME és/vs Android

Nem ördöngös megtanulni, lehet vele villogni. ☺



AMI KIMARADT

J2ME



AMI SZINTÉN KIMARADT

És még sokan mások…

AMI NEM MARAD KI

1. Gyakorlati zh – dec. 14. (csütörtök), 12:55

2. Elméleti zh – dec. 21. (csütörtök), 11 óra, 400 fős

3. Vizsga