Contents

1. Continutul cursului

Java language, platform
.NET platform, C# language essentials
Generics, Collections Java, C#
IO Java, C#
Lambda, streams Java, C#
Reflection C5 – Java
XML – Java
GUI -JavaFX - Java
Concurrency – Java, ~~C#~~
Delegates, LINQ – C#

Sabloane de proiectare folosite la acest curs: Singleton, Factory Method, Factory, Strategy, Template Method, Observer, Command, GRASP.

2. Evaluarea

2.1 Pe parcursul semestrului

- NotaLaborator=30% din NotaFinala, unde 20% din NotaLaborator reprezinta lucrarea din saptamana 10, iar 80% reprezinta media ponderata a notelor la cele 7 teme de laborator, avand ponderile 2 2 2 2 1 4 2.

2.2 In sesiunea de examene, avand datele 23, 24, 25, 26 Ianuarie

Preconditie sustinere examen: *Pentru ca un student sa se poata prezenta la examen in sesiunea de examene trebuie sa indeplineasca simultan urmatoarele 3 conditii:*

1. *Maxim **2** absente la laborator*
2. *Maxim **4** absente la seminar*
3. *Minim **4.5** la NotaLaborator.*

Important: Daca una dintre cele 3 conditii de mai sus nu este indeplinita, studentul va putea sustine examenul la disciplina MAP doar in sesiunea de restanta (23 februarie). Exceptie de la aceasta regula fac studentii din anul 3 sau din prelungire.

În sesiunea de examene, fiecare student va sustine un examen scris si unul practic, notele la aceste examene avand urmatoarele ponderi din NotaFinala (nota la disciplina MAP).

- NotaExamenScris=30% din NotaFinala
- NotaExamenPractic=40% din NotaFinala

Examenul scris

Durata: 45 minute (8:00- 8:45)

E necesar ca nota la examenul scris sa fie cel putin 5 pentru a putea promova la disciplina MAP.

Examenul practic

Durata: 2 ore si 15 de minute (10:15 – 12:30)

E necesar ca nota la examenul scris sa fie cel putin 5 pentru a putea promova la disciplina MAP.

- Fiecare student are dreptul sa se prezinte la un examen de doua ori: in prima sesiune, denumita **sesiune de examene** si in cea de-a doua sesiune, **denumita sesiune de restante sau de mariri ☺.**
- Pentru sesiunea de examene, fiecare grupa are alocata doua date. Trebuie sa veniti la examenul din sesiunea de examene la prima data alocata pentru grupa voastra.
    - Data secundara este doar pentru cazuri exceptionale.
    - Daca va fi cazul sa veniti la data secundara, trebuie sa trimiteti un mail cu motivarea cererii inainte cu cel putin 48 de ore.

2.3 In sesiunea de restante (mariri)

- Cei care doriti sa veniti la marire de nota, va rog sa-mi scrieti un mail, pana cel tarziu la data de 19 februarie 2018!

- In sesiunea de restante (mariri) evaluarea este la fel cu cea din sesiunea normala, atat pt cei care nu au promovat in sesiunea de examene sau nu s-au putut prezenta in sesiunea de examene cat si pt cei care vin la marire de nota.

- In sesiunea de restante nu se mai pot preda laboratoare, fiind considerata activitate pe parcursul semestrului!
- Notele mai mari decat 5 la examenul practic sau la examenul scris din sesiunea normala, pot fi recunoscute daca studentul doreste acest lucru. Deci, poate sustine doar examenele nepromovate din sesiunea de examene.

3. Datele de examinare si alte aspecte referitor la examen

Vezi planificarea examenelor de pe pagina facultatii: http://www.cs.ubbcluj.ro/planificarea-sesiunii-de-examene-an-univ-2017-2018-sem-i/

Consultati frecvent planificarea, e posibil sa mai apara modificari!

Important:

- Planificati-va timpul astfel incat sa ajungeti la examen inainte cu minim 10 minute inainte de ora inceperii.
- Trebuie sa aveti la voi un act de identitate (buletin, pasaport, carnet de student)

4. Examen Scris 45 minute

- Intrebari scurte care vizeaza exemplificarea unor concepte teoretice (OOP, principii de proiectare, clase interne, interfete functionale, referinta la functii, interfete functionale predefinite, reflection, delegates, events, extensions methods, synchonization, responsive GUI,)
- Mici exemple de cod si intrebari de genul ce afiseaza urmatorul program, sau completati urmatorul program....

4.1 Exemple grile:

1. Ce se intampla cand se incearca compilarea si rularea codului urmator? Explicati si, daca este cazul, corectati.

```
class A{
    int i;
    public A(int i){
        System.out.print("A()");
    }
}
class B extends A{
    float i = 3;
    public B(){
        System.out.print("B()");
    }
}
public class AB extends B{
    public static void main(String argv[]){
        Arrays.asList(new AB(), new AB(), new AB())
                .forEach(x->x.testMethodCA("AB"));
    }
    public void testMethodCA(String s){
        System.out.println(s);
    }
}
```

2. Care linii de cod din corpul interfeței JavaInterface (numerotate de la 1 la 7) sunt corecte?

```
interface JavaInterface {
    protected int x = 10;
    int y;
    int z = 20;
    default
    int x(){ return 0;}
    abstract void foo();
    final int f(int x);
}
```

3. Ce se intampla cand se incearca compilarea si rularea codului urmator? Explicati si, daca este cazul, corectati.

```
class AAA {
    class BBB {}
    static class CCC{}
```

4. Vrem să implementăm un framework de user interface. Cu ce design pattern am putea modela

}
public class Ex4_224 {
public static void main(String[] args) {
// in main
AAA a = new AAA();
AAA.BBB b=a.new BBB();
AAA.CCC c= new AAA.CCC();
}
}
	comportamentul de onClick → doSomething pentru un element de tip buton oarecare?
	
	1. Command
	2. Singleton
	3. Factory
	4. Observer

4.2 Exemple teoretice

1. Tratarea exceptiilor in C#. Definirea exceptiilor utilizator. Exemple.
2. Metode delegate in C#. Multicast delegate. Exemple
3. Modificatorul final in Java.
4. Extensii de metode in C#
5. Comentati si explicati prin exemple urmatoarea euristica OOP: "Favorizati compunerea in locul mostenirii".
6. Adevarat sau Fals:
   6.1 *O clasa poate fi declarata abstracta fara a avea metode abstracte.*
   6.2 *Orice data membru declarata in interfata este implicit* ***public****,* ***static, final***
   6.3

5 Examen Practic 2 ore si 15 minute

5.1 Java

- Va fi o problema mai complexa, de genul laboratorului, dar cu mai putine functionalitati, de genul statistici, rapoarte....
- Nu se va cere sa implementati toate operatiile CRUD pt persistenta unei entitati.
- Datele se citesc din fisiere text al carui format se da.

URMARIRE BUG-URI

O firma producatoare de software pune la dispozitie programatorilor si verificatorilor (testers) un sistem prin care acestia pot sa comunice electronic. Astfel, fiecare dintre angajatii de mai sus are un terminal(fereastra) prin care:

• verificatorul **poate înregistra un bug**, dându-i o denumire si o descriere; imediat dupa înregistrarea bug-ului, toti programatorii vad lista bug-urilor actualizata cu aceasta valoare noua introdusa;

• programatorul vizualizeaza lista bug-urilor; de asemenea**, programatorul poate selecta un bug din lista si poate declansa un buton prin care declara ca bug-ul a fost eliminat**, caz în care bug-ul este scos din lista tuturor programatorilor.

Creati 2 ferestre pentru programatori si una pentru tester si simulati actiunile prezentate mai sus.

C#

- Va fi o problema mai putin complexa decat la Java, care se poaterezolva in max 30 de minute.
- Datele se citesc din fisiere text sau xml (LINQ – sem 12)

Alte informatii

- Sunteti incurajati sa veniti cu laptopurile voastre!