FIŞA DISCIPLINEI

Anul universitar 2015-2016

Decan,

.....

1. Date despre program

1.1 Instituţia de învăţământ superior	Universitatea Tehnică „Gheorghe Asachi” din Iaşi
1.2 Facultatea	Automatica si Calculatoare
1.3 Departamentul	Calculatoare
1.4 Domeniul de studii	Calculatoare si Tehnologia Informatiei
1.5 Ciclul de studii[1]	Licenta
1.6 Programul de studii	Tehnologia informației

2. Date despre disciplină

2.1 Denumirea disciplinei			Paradigme de Programare				
2.2 Titularul activităţilor de curs			Mihai Horia Zaharia				
2.3 Titularul activităţilor de aplicaţii			Mihai Horia Zaharia, Adrian Alexandrescu				
2.4 Anul de studii[2]	2	2.5 Semestrul[3]	4	2.6 Tipul de evaluare[4]	Ex	2.7 Tipul disciplinei[5]	DID

3. Timpul total estimat al activităţilor zilnice (ore pe semestru)

3.1 Număr de ore pe săptămână	5	din care 3.2 curs	3	3.3a sem.	-	3.3b laborator	2	3.3c proiect	-
3.4 Total ore din planul de învăţământ[6]	70	din care 3.5 curs	42	3.6a sem.	-	3.6b laborator	28	3.6c proiect	-

Distribuţia fondului de timp[7]	Nr. ore
Studiul după manual, suport de curs, bibliografie şi notiţe	30
Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate şi pe teren	10
Pregătire seminarii/laboratoare/proiecte, teme, referate şi portofolii	20
Tutoriat[8]	10
Examinări[9]	4
Alte activităţi:	-

3.7 Total ore studiu individual[10]	74
3.8 Total ore pe semestru[11]	144
3.9 Numărul de credite	6

4. Precondiţii (acolo unde este cazul)

4.1 de curriculum[12]	• CTI.DF.105, CTI.DF.106
4.2 de competenţe	• CTI.DF.111

5. Condiţii (acolo unde este cazul)

5.1 de desfăşurare a cursului[13]	• Videoproiector + tabla
5.2 de desfăşurare a seminarului / laboratorului / proiectului[14]	• Computer si pachetele software coform programei

6. Competenţele specifice acumulate[15]

		Număr de credite alocat disciplinei[16]: **6**	Repartizare credite pe competenţe[17]
Competenţe profesionale	CP1	Operarea cu fundamente ştiinţifice, inginereşti şi ale informaticii	0.6
	CP2	Proiectarea componentelor hardware, software şi de comunicaţii	1.2
	CP3	Soluţionarea problemelor folosind instrumentele ştiinţei şi ingineriei calculatoarelor	1.8
	CP4	Proiectarea şi integrarea sistemelor informatice utilizând tehnologii şi medii de programare	0.6
	CP5	Întreţinerea şi exploatarea sistemelor hardware, software şi de comunicaţii	1.5
	CP6	Utilizarea sistemelor inteligente	-
Competenţe transversale	CT1	Comportarea onorabilă, responsabilă, etică, în spiritul legii pentru a asigura reputaţia profesiei	0.3
	CT2	Identificarea, descrierea şi derularea proceselor din managementul proiectelor, cu preluarea diferitelor roluri în echipă şi descrierea clară şi concisă, verbal şi în scris, în limba română şi într-o limbă de circulaţie internaţională, a rezultatelor din domeniul de activitate	-
	CT3	Demonstrarea spiritului de iniţiativă şi acţiune pentru actualizarea cunoştinţelor profesionale, economice şi de cultură organizaţională	-

7. Obiectivele disciplinei (reieşind din grila competenţelor specifice acumulate)

7.1 Obiectivul general al disciplinei	• Prezentarea principalelor paradigme de programare care domina dezvoltarea sistemelor software la ora aceasta si deprinderea de abilitati concrete de utilizare a lor.
7.2 Obiective specifice	•

8. Conţinuturi

8.1 Curs[18]	Metode de predare[19]	Observaţii
1. Introducere in paradigme 3h a. Obiectivele cursului b. Criterii folosite la evaluarea activitatii c. Indatoririle studentului d. Introducere minimal in maniera de constructie a unui compilator e. Limbaje interpetate si maniera de executie (Java) f. Clasificari ale limbajelor dominante g. Definitii (paradigma de programare si utilitatea ei) h. Prezentarea majoritatii paradigmelor de programare existente la ora actuala impreuna cu exemple din programele care le suporta **2. paradigma structurata (ASM,C) 3h** a. structura interna a unei arhtecturi microprogramae simple si exemplu de microinstructiune b. tehnici specifice de reprezentare ale algoritmilor in ASM (schema logica) si exemple c. procesor in mod real exemplificare pe arhitectura x86 d. introducere in ASM pentru Windws/DOS (variabile, instructiuni de baza) e. echivalenta dintre intructiunile de comanda a fluxului program (if, for, while) in C si reprezentarea in asamblare f. tehnici de transfer parametri la apel intre C si asamblare (managed/unmanaged) g. introducere in asamblare sub Linux (variabile, instructiuni de baza) h. introducere in modul protejat pentru procesoarele din familia x86 **3. Paradigma orientata obiect (CPP, JAVA) 3h** a. Tehnici empirice de inginerie software 1. (top-down, bottom-up si combinatii) 2. Descompunere functionala b. Gandirea orientata obiect (definitie si elemente fundamentale) c. Metodologie de proiectare a unei aplicatii OOP (Brainstorming, Filtrarea, Scenarii, Algoritmi) d. Functionalitati oferite de oop (Incapsularea, Clase, Mostenirea, Polimorfism (C++, C#)) **4. Reprezentari abstracte (UML, Java) 3h** a. Definitie si utilitate b. Ciclul de viata al unei aplicatii c. Model si probleele modelarii d. Entitati specifice UML (Structurale, Comportamentale, Grupari, Aplicatii aditionale) **5. Paradigma orientata eveniment (Java) 3h** a. Definitie si forma generala b. Abordari specifice (polling, interrupt driven, event driven) c. Structura unei aplicatii orientate pe eveniment d. Programarea orientata pe eveniment (consola, GUI) e. Edp in html+script f. EDP in java (AWT,Swing)	Expunere combinata cu prelegere dar si comentarii suplimentare la tabla cu suport pe videoproiector Discutii suplimentare la cererea studentului in afara orelor aferente curs sau aborator	Ar trebui completat cu o ora de proiect o data la 2 saptamani

g. EDP in .NET h. EDP in Python i. EDP in Android j. Aplicatii Event-driven architecture (EDA) **6. Modele de proiectare (Java) 3h** a. Definitie b. Relatii intre patter-urile din GoF c. Standardul OMT d. Modele creationale e. Modele structurale f. Modele comportamentale **7. Paradigma orientata aspect 3h** a. Termeni si definitii b. Identificarea aspectelor periferice (crosscutting concerns c. Pasii in proiectarea unei aplicatii AOP (Descompunere, Implementare, Recompunere) d. Join Points (puncte de legătură) e. Pointcuts (puncte de tăiere) f. Advice g. Aspect h. Etapele aplicării AOP i. Utilizarea USE CASE in proiectarea unei aplicatii AOP **8. Paradigma Secventiala versus Concurenta 3h** a. Arhitecturi paralele b. Nivele de paralelism c. Procese (heavy and light weight) d. Thread-uri e. Paralelismul la nivelul proceselor f. Paralelismul la nivelul thread-urilor g. Architectura generala a modelului bazat pe thread-uri h. Modele de programare folosind thread-uri (master-slave, peer, pipeline) i. Planificarea proceselor/firelor de executie (ordinea executiei programului,planificatorul CPU, selectorul de procese, Criterii pentru realizarea planificarii, algoritmi de planificare (FCFS, SJF, pe prioritat, RR, cozi multinivel) j. Planificarea in Linux k. Problema inversiunii prioritatilor l. Coerenta datelor si rezolvarea ei **9. Paradigma orientata component 3h** a. Definitii b. Java Beans c. Tehnologia EJB d. Arhitectura EJB e. Arhitecuri distribuite specifice (de la 2 straturi panala n-layer:n-tier) f. DCE g. CORBA h. ACTIVEX i. Java RPC j. Java RMI k. EJB Oracle **10. Paradigma programarii vizuale 3h** a. Definitii si clasificari b. Istoric si evolutie c. VPL pure d. Orfani e. Paradigme ale programrii vizuale f. Strategii de proeictare in VPL g. Programarea functional vizuala h. Programarea vizuala prin exemple		

i. Programarea vizuala bazata pe constrangeri j. Limbaje hibride de ultima ora **11. Paradigma orientata pe expresii (Python) 3h** a. Privire de ansamblu b. Sintaxa cuvinte cheie operatori instructiuni de control,afisare c. Operatiipe liste d. Gestiue dictionary e. Lucru cu fisiere f. OOP in Python g. Gestiunea thread-urilor **12. Programarea functionala in Python 3h** a. Foobar decorator operator @ si exemple b. Modulul wrapt c. λ-calcul definitie si fundamente d. λ-calcul in Python e. Functii partiale f. Functii anonime g. Inchideri h. Scoping lexical i. Functii Lambda cu scurtcircuit j. Cicluri functionale in Python k. bucla ecou functionala vs imperativa l. Eliminarea efectelor colaterale m. Lucrul cu MySQL in Python n. Automate in Python o. Aplicatii client server p. Circuite logice in Python q. Functii one way **13. Paradigma programare generica 3h** a. Definitie si istoric b. Programarea functional c. Definitie si istoric d. Programarea fucntionala e. Formatare cod sursa f. Tipuri de date g. Functii Curry h. Functii polimorfice i. Supraincarcarea functiilor j. Clase de tip k. Expresii conditionale l. Ecuatii guarded m. Pattern matching n. Modele de liste si multimi o. Functii lambda p. Sectiuni q. Gardieni r. Functia zip s. Functii cu notatie sintetica t. Functii recursive si recursivitate u. Sortare v. Functii de nivel superior w. Primitive derivate si expresii aritmetice x. Programe interactive y. Arbori **14. Paradigma programarii logice (Prolog) 4h** a. Definitii si istoric b. Logica predicatelor si calcul predicativ (pe scurt) c. Reprezentarea cunostiintelor in Prolog d. Definia relatiilor prin fapte e. Definia relatiilor prin reguli f. Trace and Notrace		

g. Cum raspunde Prolog la intrebari h. Operatii itipice Prolog i. Intelesul declarativ al programelor de tip Prolog j. Simularea unui automat nedeterminist finit k. Comunicarea cu fisierele		

Bibliografie curs:

1. http://www.cs.gsu.edu/~cscnxx/index-2310.html
2. http://www.december.com/html/x1
3. http://www.cs1graphics.org/
4. David Luckham, **The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems Hardcover ,** Addison-Wesley Professional, 2002
5. http://www.complexevents.com/
6. http://dist.codehaus.org/esper/NYJavaSIG_May_30_2007.pdf
7. http://ooad.se.sjtu.edu.cn/ppt/Chap%206%20-%20Design%20Patterns.ppt
8. http://www.vincehuston.org/dp/
9. http://www.dre.vanderbilt.edu/~schmidt/LiveLessons/#JavaSourceCode
10. http://java.dzone.com/articles/design-patterns-bridge
11. http://www.yaldex.com/java_tutorial/0939352869.htm
12. http://triplexkry.blogspot.ro/2011/06/uml-use-case-diagrams.html
13. http://www.ivarjacobson.com/Use_Case2.0_ebook/
14. http://www.omgwiki.org/architecture-ecosystem/doku.php?id=composite_concepts
15. https://www.cs.umd.edu/class/spring2006/cmsc132/
16. http://people.cs.uchicago.edu/~asiegel/courses/cspp51037/
17. http://www.cis.upenn.edu/~lee/01cis642/
18. http://users.ece.gatech.edu/~copeland/jac/3055-05/
19. http://www.buyya.com/
20. http://lass.cs.umass.edu/~shenoy/courses/spring05/
21. http://www.jguru.com/faq/view.jsp?EID=143462
22. http://docs.oracle.com/javase/specs/jls/se7/html/jls-17.html
23. http://docs.oracle.com/javase/tutorial/essential/concurrency/exinter.html
24. http://java2all.com/technology/core-java/multithreading/ thread-life-cycle
25. ANDY JU AN WANG KAI QIAN, ***COMPONENT-ORIENTED PROGRAMMING***, willey, 2005
26. http://bigfoot.cs.upt.ro/~ioana/arhit/
27. http://www.cs.fsu.edu/~awang/
28. http://www.cs.ubbcluj.ro/~florin/PJ/
29. http://docs.oracle.com/javaee/6/tutorial/doc/
30. http://cs.unitbv.ro/site/pagpers/scheiber/html/distribuit.xhtml
31. Deepak Vohra , EJB 3.0 Database Persistence with Oracle Fusion Middleware 11g, 2010 ISBN: 1849681562
32. http://www.byte.ro/byte97-04/08jbeans.html
33. http://www.tutorialspoint.com/jsp/jsp_java_beans.htm
34. Crimi, C., A. Guercio, G. Pacini, G. Tortora, and M. Tucci, "Automating Visual Language Generation," IEEE Transactions on Software Engineering, vol. 16, no. 10, pp. 1122-1135, October 1990
35. Margaret M. Burnett and Herkimer J. Gottfried. 1998. Graphical definitions: expanding spreadsheet languages through direct manipulation and gestures. *ACM Trans. Comput.-Hum. Interact.* 5, 1 (March 1998), 1-33.
36. http://www.cs.washington.edu/homes/jpower/vpl/vpl_home.html
37. http://www.wi.leidenuniv.nl/CS/SEIS/vislang/VLcourse.html
38. http://www-lsi.upc.es/~rbaeza/cursos/vp/todo.html
39. http://www.cs.berkeley.edu/~maratb/cs263/
40. http://www.efd.lth.se/~d87man/EXJOBB/MSC.html
41. http://www.ecs.soton.ac.uk/~tal00r/vlangs.html
42. http://www.researchgate.net/publication/236132854_Demo_Abstract_SEAL-Blockly_Sensor_Network_Visual_Programming_Using_a_Web_Browser/file/9c960516422aea7b94.pdf
43. http://www.schiffer.at/vp/html/liter.htm
44. Erwig, M. & Schneider, M. (2000). Query-By-Trace: Visual Predicate Specification in Spatio¬Temporal Databases. In Arisawa, H. and Catarci, T., editors, Advances in Visual

Information Management - Visual Database Systems, pages 199-218. Kluwer Academic Publishers, Boston, MA.
45. http://www.ickn.org/elements/hyper/cyb105.htm
46. http://www.clarity-support.com
47. Johnston, W.M.; Hanna, J.R.P. and Millar, R.J. (2004). "Advances in dataflow programming languages" *ACM Computing Surveys* **36** (1): 1–34. doi:10.1145/1013208.1013209
48. https://www.iam.unibe.ch/scg/svn_repos/Lectures/ProgrammingLanguages/12VisualProgramming.ppt
49. Ivan Edward Sutherland, *Sketchpad: A man-machine graphical communication system*,
50. Ph.D. thesis, MIT, January 1963. www.cl.cam.ac.uk/techreports/UCAM-CL-TR-574.pdf
51. http://www.i3s.unice.fr/~mosser/_media/research/iawtic08.pdf
52. http://www.apple.com/macosx/features/automator/
53. www.nakedobjects.org
54. Jonathan Edwards, "No ifs, ands, or buts: uncovering the simplicity of conditionals," *OOPSLA* 2007.
55. http://subtextual.org/
56. Shi-Kuo Chang, "Visual languages: a tutorial and survey," IEEE Software, 1987
57. Gaelli, et al., *Idioms for Composing Games with EToys*, C5 2006
58. *ftp://ftp.cs.orst.edu/pub/burnett/**VPL**classification.JVLC.Sept94.pdf*
59. www.iam.unibe.ch/~scg/Teaching/CP/PetriNets
60. *http://web.engr.oregonstate.edu/~erwig/papers/*
61. "LabVIEW Graphical Programming: Practical Applications in Instrumentation and Control" by Johnson Gary W., Gary W. Johnson. McGraw-Hill, 1997.
62. Kahn and Vijay A. Saraswat in "Actors as a special case of concurrent constraint (logic) programming", in SIGPLAN *Notices*, October 1990
63. **https://thenewcircle.com/bookshelf/python_fundamentals_tutorial/functional_programming.html**
64. **http://www.ibm.com/developerworks/linux/library/l-prog3/index.html**
65. http://www-106.ibm.com/developerworks/library/l-prog2.html
66. **http://www.ibm.com/developerworks/linux/library/l-prog/index.html**
67. http://wrapt.readthedocs.org/en/latest/
68. http://docs.python.org/2/library/itertools.html
69. http://pythonconquerstheuniverse.wordpress.com/2012/04/29/python-decorators/
70. http://freepythontips.wordpress.com/2013/12/05/python-decorators-finally-demystified/
71. http://openbookproject.net/py4fun/
72. David Mertz, Text Processing in Python ,Addison Wesley,2003
73. Brad Dayley, Python Phrasebook: Essential Code and Commands,Sams Publishing,2006
74. http://www.laurentluce.com/posts/python-and-cryptography-with-pycrypto/
75. **http://www.ps.uni-saarland.de/~duchier/python/continuations.html**
76. https://pypi.python.org/pypi
77. **http://www.activestate.com/activepython**
78. **http://blog.nektra.com/main/2012/07/20/windows-api-hooking-in-python-with-deviare/**
79. **http://www.learnpython.org/**
80. **http://courses.cs.washington.edu/courses/cse142/08au/python/**
81. **http://simeonfranklin.com/blog/2012/jul/1/python-decorators-in-12-steps/**
82. **https://developers.google.com/appengine/docs/python/gettingstartedpython27/introduction**
83. Graham Hutton, Programming in Haskell", Cambridge University Press, 2007
84. http://www.scs.stanford.edu/11au-cs240h/notes/ghc-slides.html

8.2a Seminar	Metode de predare[20]	Observaţii
-	-	-
8.2b Laborator	Metode de predare[21]	Observaţii
Lucrarea 1 Task 1.Deschideti MS Visual si testati (5 minute) Task 2.Creati si executati vesnicul program HelloWorld in stil C (structurat) (5 minute) Task 3.Creati si executati/depanati in CPP o clasa HelloWorld dar cu (10 minute) - un constructor implicit care seteaza sirul de afisat ca fiind "HelloWolrd" - un constructor explicit care permite primirea la initalizare a unui	Demonstratie practica execitiu si experiment	none

sir e afisat - o metoda care afiseaza mesajul - o metoda care sterge mesajul Task 4. Folosind pointeri void si cast-ul explicit definiti, initializati, modificati, interpretati multiplu aceasi variabila fara tip 10 minute Task 5.Creati o ierarhie de clase formata din (20 minute) clasa abstracta mamifer (attribute (tip- (evil,divine), data nasterii) si metode (mananca, merge la baie, hraneste animalele) Se deriveaza o clasa bunica si clasele PisicaDeCartier, PisicaSiameza si Pisica egipteana Acestea din urma mai au metoda miauna unde se aplica polimorfismul si fiecare miauna diferit Actiunile in acest caz sunt implementate prin mesaje afisate la consola Task 6.Analizand codul de mai jos sa fie reproiectat (extragere relatii intre clase, eliminarea referintelor la apelul zone grafice si afunctiilor asociate) astfel incat sa functioneze in mod text (cerc e mai complicat dispare) dar line si ca forma sa avem patrat si dreptunghi 50 min Tema pe ACASĂ Extindeţi aplicaţia oferind suport pentru mai multe forme geometrice (pe lângă comanda de desenare în sine, nu uitaţi de opţiunea help). Pe lângă cele două forme simple deja prezentate, încercaţi sa adăugaţi altele noi: triunghi, hexagon, si pentagon. **Lucrarea 2** Task 1: Obisnuirea cu mediul de lucru MASM 10 min Task 2: Factorial pentru 286 masm (20 min) Task 3: Sa se modifice exemplul din curs astfel incat sa sepoata face calcul lui $\sum_{i=0}^{n} i$ (20 min) (hint: faceti schema logica) Task 4: Sa se implementeze programul conform schemei logice de mai jos (30 min) Task 5: Sa se implementeze ordonarea unui vector folosind swap din curs (20 min) Tema pe ACASA : (indiferent daca este sub linux sau sub MS – la alegerea clientului) sa se implementeze o are buble sort pentru nota 8 sau alt algoritm pentru nota 10 **Lucrarea 3** Testare cunostiintelor de la curs prin intrebari rapide (5 minute) Task 1 Sa se implementeze cu template o stiva(20 min) Task 2 Sa se implementeze cu template lista circulară (20 min) Task 3 Sa se implementeze cu template lista cu numere complexe si operatori supraincarcati ca in curs (20 min) Task 4 Sa se implementeze hambarul din curs, cateva reguli de interactiune intre elementele din hambar din curs, instantieri aleatoare a unui numar si tip de animale se executa/interactioneaza in clasa curte pe baza regulilor (20 min) Tema pe acasa: pornind de la model hambar curte interactiuni sa se proiecteze un joc cu nave in spatiu (navele diferite(putere de foc, rezistenta la atac, viteza, manevrabilitate etc) in loc de actiuni grafice se vor afisa mesaje descriptive a ceea ce ar trebui sa se vada/intample grafic **Lucrarea 4**		

Sa se implementeze aplicatia ceasului discutata la curs care are a. urmatoarea diagram de clase: b. functionalitatea descrisa mai jos c. urmatoarea diagrama de secventa d. si urmatoarea diagram de stare (50 min) Pornind de la urmatoarea diagram de stare care se refera la gestiunea inscrierilor la un curs optional pentru oricare profesor dintr-un department Tema acasa: Sa se scrie o aplicatie care defineste o clasa Numarator care la schimbarea valorii numaratorul genereaza un eveniment CounterEvent. Acest eveniment va fi receptat de o clasa Receptor care la aparitia evenimentelor afiseaza valoarea numaratorului. Pentru orice eveniment nou se creaza o clasa derivata din clasa EventObject, iar interfetele pentru receptionarea evenimentelor trebuie sa extinda clasa EventListener din pachetul java.util. Clasa Counter este un numarator care se ruleaza pe un fir de executie propriu si este capabil de gestionarea obiectelor de tip listener. Aceste obiecte listener sunt stocate intr-un vector (java.util.Vector) si la modificarea numaratorului sunt anuntati. (notifyCounterChange). **Lucrarea 5** Evenimente in AWT Vom incepe prin a prezenta doua tabele care sunt utile pana se retin asocierile intre containere, component si evenimente Tema 1. Sa se scrie o aplicatie care afiseaza o fereastra avand o caseta text si trei butoane avand etichetele 1,2,3. La apasarea butoanelor in caseta text sa apara eticheta butonului apasat si care are diagrama de clasa de mai jos (20 min): Tema 2. Sa se scrie o aplicatie care se ruleaza intr-o fereastra si la apasarea mouseului in zona ferestrei sa apara o eticheta care sa contina coordonatele apasarii mouseului sic area are Diagrama de clasa de mai jos (20 min) Tema 3. Sa se afiseze un cerc intr-un canvas (10 min) II. Utilizarea componentelor Swing Tema 4. Sa se Creeze un editor de tip notepad (avem unde edita, bare de scrol, butoane control, etc) 50 min. Tema pe acasa. Sa se proiecteze (clase creion hartie etc) si sa se implementeze intefata grafica a jocului cu bataie in spatiu de ora trecuta Va contine o zona de desenare (unde se va afisa pentru test o nava piramidala desenata), lista de selectie tip de nava, layout-urile vor fi functie de imaginatia fiecaruia dar trebuie, butoane de comanda si control, zona grafica separata de afisare a pozitiei spatiale globale pentru toate navele, cutie pt preluarea datelor unde preiau numele jucatorului, afisare coordonate, etc) **Lucrarea 6** Tema 1 Fie fabrica de forme prezentata in curs: Pornind de la aceasta sa se construiasca doua fabrici de obiecte (50 min) - fabrica de nave spatiale una cub, una piramidala si una sferica care sa fie desenate pe canvas. Navele sa primeasca si un parametru pentru culoarea de desenare - o fabrica de “foc” in care sa se obtina trei tipuri de foc pentru navele antarioare linear – mitraliera, blast – o minge de foc si rachete - mai multi clindri. Tema 2 API-ul Java oferă două interfeţe pentru a uşura		

implementarea acestui şablon: Obsever şi Observable. Exemplul următor arată cum pot fi folosite aceste interfeţe. Pornind de la acesta sa se implementeze un observer general pentru navele spatiale anterioare care sa permita ca fiecare nava sa primeasca informatii complete despre celelalte nave (viata, foc, directie foc, tip foc, putere nava, clasa navei etc) (50 minute)
Tema acasa: Pornind de la exemplul din curs sa se creeze un adapter pentru fiecare din tipurile de functii pentru foc ale navelor asfel incat oricare nava sa poata sa foloseasca oricare tip de foc (cand face upgrade de exemplu)

Lucrarea 7
Exemplu de program AspectJ
Exemplu caching pentru un algoritm de calcul factorial
Sa reanalizam problema hotelului. Din punct de vedere al componentelor (clase, module) în care se poate diviza sistemul ce trebui implementat, distingem următoarele (nivel minimal):
• Customer şi HotelStaff: actorii sistemului.
• Room şi Reservation: datele cu care aceştia operează.
• ReserveRoom, CheckInCustomer, CheckOutCustomer: operaţiile ce pot fi efectuate de către actori, ale căror rezultate sunt vizibile în stările datelor.
Pentru început stabilim clasele de bază, care conţin informaţii ce nu sunt legate în mod explicit de o anumită operaţie.
Tema 1. Creaţi o aplicaţie de test care să permită simularea celor 3 operaţii de bază implementate de către sistem.Identificaţi elementele cheie ce trebuie extinse pentru ca aplicaţia sa fie cât mai realistă (ex. la rezervare clientul alege şi perioada, după care se calculează un preţ pe acea perioadă) (50 min).
Tema 2. Odată realizată această implementare, gândiţi o modalitate de a extinde o funcţionalitate: de ex, atunci când nu se găsesc camere de tipul dorit libere, să se realizeze o listă de aşteptare. Această extra funcţionalitate trebuie implementată în mod identic cum au fost implementate cele de bază, adică fără a modifica codul deja existent (inclusiv aspectele).(50 min)
Tema acasa: Creati un aspect peste proiectul cu navele care sa permita modificarea comportamentului unei nave (de exemplu inainte de a trage un foc(cand este apelat focul) sare inainte cu o cuanta (spre adversar) si dup ace a tras face un salt in lateral cu o cuanta spatiala)

Lucrarea 8
Reamintim cateva elemente din curs :
1. Un exemplu de clasa imutabila
2. Sincronizarea accesului la campuri “mutable”
3. Corectati codul de mai jos pentru a-l putea folosi mai departe in teme
4. Exemplu incomplet de pool

Tema 1. Sa se implementeze un program Java care va implementa un scheduler pentru thread pool (ferma de cai pardon de fire de executie) in conformitate cu algoritmul FCFS prezentat la curs)
Tema 2. Pornind de la exemplul de buffer inplementat cu producator consummator din curs (cu eliminarea blocajului) sa se implementeze o coada folosita In transmiterea si receptia de siruri de caractere intre mai multhe thread-uri (un fel de chat unde fiecare client este pe cate un thread)
Tema 3. sa se construiasca un pool de thread-uri care sa faca niste calculi simple (gen sume, inductie etc) sa puna in evidenta hazardul de curse (vez la in curs)
Tema 4. Sa se implementeze un calcul $\sum_0^n i$ cu 4 thread-uri

simultane care fiecare calculeaza pe un subinterval folosind modelul master/slave

Tema 5 sa se proceseze calcul $\sum_0^n i$ simultan pentru 4 valori diferite a lui n luate dintr-o coada de catre 4 taskuri diferite (model peer)

Tema 6. Sa se realizeze o procesare dupa model pipeline a unui tablou de intregi. Primul thread din pipe inmulteste toate elemental vector V cu o constanta alpha, urmatorul thread din pipe va ordona vectorul iar final ultimul thread il va afisa in coordinate x si y

Tema 7. Sa se analizeze exemplul de mai jos si sa se puna in evidenta situatiile de lock always, never, sometimes si hostile daca este cazul. Sistemul are evitat blocajul? De ce?

Tema 8. Sa se implementeze mai multe cozi de thread-uri cu prioritati diferite si sa se implemnteze mecanismul de prevenire a infomatarii (cel de imbatranire)

Tema 9. Pentru sincronizarea unor thread-uri sa se implemnteze protocolul cu simularea limitarii prioritatii (Highest locker)

Tema 10. Sa se scrie un program simplu cu thread-uri (pornind de la exemplul din curs) care sa foloseasca atat lock-ul pe instanta cat si cel static

Tema 11. Sa se modifice/corecteze programul de mai jos pentru a introduce inca o clasa C pastrand idea generala

Tema pe acasa: Fiecare nava din joc este un thread dintr-un pool. Thread-urile de nave au prioritati diferite si de aceea cand comunica (vezo observer etc) trebuie utilizat algoritmul highest locker

Lucrarea 9

Instalarea si configurarea mediului de dezvoltare (20 min

Creearea aplicaţiei EJB

Tema 1. Testaţi urmatoarea aplicaţie EJB (30 min).

Tema 2. Creaţi o aplicaţie EJB care să permită gestionarea cărţilor dintr-o bibliotecă pornind de la următoarele interfeţe (40 min):

Tema 3. Creaţi o aplicaţie client pentru a testa aplicaţia EJB de mai sus (10 min).

Tema pe acasa. Adăugaţi metode care să permită:

- căutarea unei cărţi după isbn,
- afişarea tuturor cărţilor scrise de un anumit autor sau publicate într-un anumit an,
- împrumutarea unor cărţi precum şi returnarea lor,
- scoaterea unor cărţi din bibliotecă,
- creaţi un meniu care să permită selectarea operaţiilor enumerate mai sus.

Lucrarea 10

Aplicatii LabView

Tema 1. Să se proiecteze un instrument virtual care face suma si diferenta a 2 numere si le afiseaza analogic.

Tema 2. Să se proiecteze un instrument virtual care calculează suma si diferenta a 2 numere reale.

Tema 3. Să se creeze un instrument virtual care să converseze o temperatură în grade Celsius într-o temperatură
in grade Farenheit. (TF=9/5*TC+32)

Tema 4. Sa se creeze un instrument virtual care sa $S = n_0 + \sum_{i=0}^{N-1} i$ pana cand 50 >S>20.
Sa se afiseze suma si valoarea contorului I.

Tema acasa. Să se creeze un instrument virtual care să implementeze ecuaţia:

$y(x) = x^2 * e^x + \sqrt{x} * \ln x + \sin x * \cos x$

Lucrarea 11
Instalare mediu
Testare exemple
Tema 1. sa se creeze un joc de tip spanzuratoarea folosinf functiile de desenare grafica in python
Tema 2. Sa se implementeze mai multe cozi de thread-uri cu prioritati diferite si sa se implemnteze mecanismul de prevenire a infomatarii (cel de imbatranire)
Tema 3. Pentru sincronizarea unor thread-uri sa se implemnteze protocolul cu simularea limitarii prioritatii (Highest locker)
Tema 4. sa se construiasca un pool de thread-uri care sa faca niste calculi simple (gen sume, inductie etc) sa puna in evidenta hazardul de curse (vez I in curs)
Tema 5. Sa se implementeze un calcul $\Sigma_0^n i$ cu 4 thread-uri simultane care fiecare calculeaza pe un subinterval folosind modelul master/slave
Tema 6 sa se proceseze calcul $\Sigma_0^n i$ simultan pentru 4 valori diferite a lui n luate dintr-o coada de catre 4 task-uri diferite (model peer)
Tema 7. Sa se realizeze o procesare dupa model pipeline a unui tablou de intregi. Primul thread din pipe inmulteste toate elemental vector V cu o constanta alpha, urmatorul thread din pipe va ordona vectorul iar final ultimul thread il va afisa in coordinate x si y
Tema pe acasa. sa se creeze un joc cu pietre (de exemplu sa le pot muta cate una pentru a le rearanaja sa iasa imagine, numere intr-o ordine etc..)

Lucrarea 12
Pe baza exemplului de la curs (sursa modulului de simulator clc: openbookproject.net/py4fun/logic/logic.py) sa se
resolve urmatoarele teme:
Tema 1. Sa se implementeze calculul functiei din figura
Tema 2. Sa se implementeze calculul functiei din figura
Tema 3. Sa se implementeze calculul functiei din figura
Tema 4. Folosind exemplul din curs sa se implementeze urmatorul automat finit
Tema 5. Sa se creeze un program care primeste un fisier oarecare (exe, dll, doc etc) la intrare si produce intr-un fisier text cu acelasi nume un hash cu md5 peste el.
Tema pe acasa. Pe baza exemplului (folosind TCP) din curs sa se realizeze o aplicatie de chat simplu intre doi parteneri

Lucrarea 13
Cateva detalii suplimentare despre limbaj:
Testati urmatoarele probleme simple:
Este un numar prim (folosind ciurul lui Eratostene)
Duplicari
8 regine
Tema 1: Sa se implementeze calculul factorialului.
Tema 2: Sa se implementeze calculul factorial al lungimii listei.
Tema 3: Sa se sorteze un text folosind implementarea quicksort din curs.
Tema 4: Sa se inverseze o lista cu foldr.
Tema 5: Sa se scrie un parser care verifica daca un numar este numar.
Tema 6: Sa se citeasca un text si sa se puna cuvintele intr-un arbore binar. Apoi sa fie folosia pe post de
dictionary.

Tema acasa: scrieti o aplicatie simpla pentru parser (vezi parser.hs si parsing.hs) **Lucrarea 14** Fie urmatorul exemplu care determinarea factorilor primi pentru un numar de intrare pozitiv:Sa se modifice astfel incat sa putem obtine numere prime dintr-un interval dat 2. Fie urmatorul exemplu de construire a codului GRAY Sa se genereze si sa se scrie intr-un fisier codul gray pentru 256 de biti (cate un cod pe linie) 3. In prolog un arbore vid este reprezentat prin atomul 'nil' si arborele nevid prin termenul t(X,L,R), unde X – radacina, L – subarbore stang, Y – subarbore drept. De exemplu pentru arborele din figura putem scrie T1 = t(a,t(b,t(d,nil,nil),t(e,nil,nil)),t(c,nil,t(f,t(g,nil,nil),nil))) si sa se testeze cu acest arbore exemplul de mai jos care extrage frunzele dintr-un arbore si le pune intr-o lista Tema 4. Sa se scrie un program Prolog care sa poata gasi ultimul element dintr-o lista (?- my_last(X,[a,b,c,d]). X = d) Tema 5. Sa se scrie un program Prolog care sa poata elimina duplicatele dintr-o lista fara a schimba ordinea acestora (?- compress([a,a,a,a,b,c,c,a,a,d,e,e,e,e],X). □ X = [a,b,c,a,d,e]) Tema 6. Sa se scrie un program Prolog care sa poata elimina fiecare al n-lea element dintr-o lista (?- drop([a,b,c,d,e,f,g,h,i,k],3,X). □ X = [a,b,d,e,g,h,k])		
8.2c Proiect	Metode de predare[22]	Observaţii
-	-	-

Bibliografie aplicaţii (seminar / laborator / proiect):

Cursul este principala sursa dar mai sunt si urmatoarele
http://www.ic.unicamp.br/~meidanis/courses/mc336/problemas-prolog/
Ivan Bratko, Programming for Artificial Intelligence, Three edition 2001, Addision Wesley
http://www.haskellforall.com/2013/12/equational-reasoning.html
http://en.wikibooks.org/wiki/Haskell/List_processing
http://www.vex.net/~trebla/haskell/lazy.xhtml
http://dannynavarro.net/2014/03/17/an-opinionated-importing-style-for-haskell/
http://pragprog.com/magazines/2012-12/web-programming-in-haskell
http://urchin.earth.li/~ian/style/haskell.html
http://sdg.csail.mit.edu/projects.html
https://docs.python.org/2/c-api/init.html#threads
http://dabeaz.blogspot.ro/2010/02/revisiting-thread-priorities-and-new.htm
documentatia de la national instruments cu privire la labview
http://www.cplusplus.com/doc/papers/ascii.html

9. Coroborarea conţinuturilor disciplinei cu aşteptările reprezentanţilor comunităţii epistemice, asociaţiilor profesionale şi angajatori reprezentativi din domeniul aferent programului[23]

- Materia pune bazele pentru urmatoarele cursuri de nivel superior: ingineria programarii, inteligenta artificiala, proiectarea translatoarelor, limbaje formale, sisteme distribuite, dezvoltarea aplicatiilor web
- Materia ofera ajutor pentru materiile de profil software dar si pentru cele de proiectare ale sistemelor hardwre de nivel 1 si doi
- Materia creaza legaturi directe/imediate cu umartoarele discipline structuri de date, proiectarea algoritmilor, sisteme de operare, analiza si sisteza dispozitivelor digitale, programarea OOP
- Cu privire la utilitatea pentru piata muncii: Aplicatile de laborator sunt 50% bazate pe limbajul Java si tehnlogii emergente acestuia deci ofera cunostiinte suficiente pe acesta directie. De asmemenea sunt utilizate si o serie de limbaje formale sau dedicate pentru proiectarea aplicatiilor web ca Python. Acesta din urma este folosit si pentru dezvoltarea de aplicatii pentru sisteme embeded

10. Evaluare

Tip activitate	10.1 Criterii de evaluare	10.2 Metode de evaluare	10.3 Pondere din nota finală
10.4 Curs	• Cunoştinţe teoretice însuşite (cantitatea, corectitudinea, acurateţea)	Teste pe parcurs[24]: 0	0%
		Teme de casă: 13	25%
		Evaluare finală:	50% (minim 5)
10.5a Seminar	• Frecvenţa/relevanţa intervenţiilor sau răspunsurilor	• Evidenţa intervenţiilor, portofoliu de lucrări (referate, sinteze ştiinţifice)	0%
10.5b Laborator	• Cunoaşterea aparaturii, a modului de utilizare a instrumentelor specifice; evaluarea unor instrumente sau realizări, prelucrarea şi interpretarea unor rezultate	• Răspuns oral • Demonstraţie practică	25% (minim 5)
10.5c Proiect	• Calitatea proiectului realizat, corectitudinea documentaţiei proiectului, justificarea soluţiilor alese	• Autoevaluarea, prezentarea şi/sau susţinerea proiectului • Evaluarea critică a unui proiect	0% (minim 5)
10.5d Alte activităţi[25]	•	•	0% (minim 5)
10.6 Standard minim de performanţă[26] cf CT1			
• Dobandirea unei gandiri analitice si a unor abilitati de generalizare independente de limbajul de programare folosit care sunt o componenta fundamentala proiectarii oricarei aplicatii software complexe. • De asemnea se dobandesc si o serie de abilitati practice legate de particularizarea paradigmelor de programare folosind o gama larga de limbaje de programare multiparadigma			

Data completării, Semnătura titularului de curs, Mihai Horia Zaharia, Semnătura titularului de aplicaţii, Mihai Horia Zaharia

22.09.2014 ....

Alexandrescu Adrian

.....

Data avizării în departament, Director departament, Petru Cascaval

25.09.2014 .....

[1] *Licenţă / Master*
[2] *1-4 penrtru licenţă, 1-2 pentru master*
[3] *1-8 pentru licenţă, 1-3 pentru master*
[4] *Examen, colocviu sau VP A/R – din planul de învăţământ*
[5] *DF - disciplină fundamentală, DID - disciplină în domeniu, DS – disciplină de specialitate sau DC - disciplină complementară - din planul de învăţământ*
[6] *Este egal cu 14 săptămâni* x *numărul de ore de la punctul 3.1 (similar pentru 3.5, 3.6abc)*
[7] *Liniile de mai jos se referă la studiul individual; totalul se completeaza la punctul 3.7.*
[8] *Între 7 şi 14 ore*
[9] *Între 2 şi 6 ore*
[10] *Suma valorilor de pe liniile anterioare, care se referă la studiul individual.*
[11] *Suma dintre numărul de ore de activitate didactică directă (3.4) şi numărul de ore de studiu individual (3.7); trebuie să fie egală cu numărul de credite alocat disciplinei (punctul 3.9)* x *24 de ore pe credit.*
[12] *Se menţionează disciplinele obligatoriu a fi promovate anterior sau echivalente*
[13] *Tablă, vidoproiector, flipchart, materiale didactice specifice etc.*
[14] *Tehnică de calcul, pachete software, standuri experimentale, etc.*
[15] *Competenţele din Grilele G1 şi G1bis ale programului de studii, adaptate la specificul disciplinei, pentru care se repartizează credite (www.rncis.ro sau site-ul facultăţii)*
[16] *Din planul de învăţământ*
[17] *Creditele alocate disciplinei se distribuie pe competenţe profesionale şi transversale în funcţie de specificul disciplinei*
[18] *Titluri de capitole şi paragrafe*
[19] *Expunere, prelegere, prezentare la tablă a problematicii studiate, utilizare videoproiector, discuţii cu studenţii (pentru fiecare capitol, dacă este cazul)*
[20] *Discuţii, dezbateri, prezentare şi/sau analiză de lucrări, rezolvare de exerciţii şi probleme*
[21] *Demonstraţie practică, exerciţiu, experiment*

[22] *Studiu de caz, demonstraţie, exerciţiu, analiza erorilor etc.*
[23] *Legătura cu alte discipline, utilitatea disciplinei pe piaţa muncii*
[24] *Se va preciza numărul de teste şi săptămânile în care vor fi susţinute.*
[25] *Cercuri ştiinţifice, concursuri profesionale etc.*
[26] *Se particularizează la specificul disciplinei standardul minim de performanţă din grila de competenţe a programului de studii.*