FIŞA DISCIPLINEI

1. Date despre program

1.1 Instituţia de învăţământ superior	**Universitatea “Alexandru Ioan Cuza” din Iaşi**
1.2 Facultatea	**Facultatea de Informatică**
1.3 Departamentul	**Informatică**
1.4 Domeniul de studii	**Informatică**
1.5 Ciclul de studii	**Licență**
1.6 Programul de studii / Calificarea	

2. Date despre disciplină

2.1 Denumirea disciplinei			**Ingineria programării**				
2.2 Titularul activităţilor de curs			**Iftene Adrian, Moruz Alex**				
2.3 Titularul activităţilor de seminar			**Iftene Adrian, Moruz Alex, Pistol Ionuț, Arusoaiei Andrei, Olariu Florin**				
2.4 An de studiu	**2**	**2.5** Semestru	**2**	**2.6** Tip de evaluare	**M**	**2.7** Regimul disciplinei*	**OB**

** OB – Obligatoriu / OP – Opţional*

3. Timpul total estimat (ore pe semestru şi activităţi didactice)

3.1 Număr de ore pe săptămână	**4**	din care: **3.2** curs	**2**	**3.3** laborator	**2**
3.4 Total ore din planul de învăţământ	**56**	din care: **3.5** curs	**28**	**3.6** laborator	**28**
Distribuţia fondului de timp					ore
Studiu după manual, suport de curs, bibliografie şi altele					**16**
Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate şi pe teren					**16**
Pregătire seminarii/laboratoare, teme, referate, portofolii şi eseuri					**16**
Tutoriat					
Examinări					**12**
Alte activităţi					**1**

3.7 Total ore studiu individual	**48**
3.8 Total ore pe semestru	**117**
3.9 Număr de credite	**5**

4. Precondiţii (dacă este cazul)

4.1 De curriculum	Algoritmi şi Programare. Programare orientată obiect
4.2 De competenţe	Programarea în limbaje de nivel înalt. Dezvoltarea și întreținerea aplicațiilor informatice. Proiectarea si gestiunea bazelor de date

5. Condiţii (dacă este cazul)

5.1 De desfăşurare a cursului	Sala de curs trebuie să dispună de video-proiector, conexiune la Internet și tablă pentru exemplificări
5.2 De desfăşurare a laboratorului	Sala de laborator trebuie să dispună de conexiune la Internet și tablă pentru exemplificări. Studenții au nevoie de calculatoare pe care să fie instalat un limbaj ce le permite să programeze orientat-obiect (Java, C#, Python, PhP, C++, etc.)

6. Competenţe specifice acumulate [definitiile conceptelor de mai jos se gasesc la adresa http://docis.acpart.ro/uploads/Fisiere/Metodologie%20CNCIS.pdf]

Competenţe profesionale	**C1. Programarea în limbaje de nivel înalt.** **C2. Dezvoltarea și întreținerea aplicațiilor informatice.** **C3. Proiectarea si gestiunea bazelor de date.**
Competenţe transversale	**CT1.** Desfășurarea eficientă a activităților organizate într-un grup inter-disciplinar și dezvoltarea capacităților empatice de comunicare inter-personala, de relaționare și colaborare cu grupuri diverse

7. Obiectivele disciplinei (din grila competenţelor specifice acumulate) la fel, detalii in documentul http://docis.acpart.ro/uploads/Fisiere/Metodologie%20CNCIS.pdf]

7.1 Obiectivul general	Construirea unei viziuni profesionale asupra procesului de dezvoltare a programelor. Studenţii vor învăţa metode şi tehnici avansate pentru realizarea unor programe complexe de o calitate superioară în contextul respectării cerinţelor clienţilor privind funcţionalitatea, costurile şi termenul limită.
7.2 Obiectivele specifice	La finalizarea cu succes a acestei discipline, studenţii vor fi capabili să: ▪ Explice cum pot fi folosite principiile programării orientate obiect în dezvoltarea proiectelor de dimensiuni mari ▪ Descrie pașii necesari implementării cu succes a unui proiect pentru un client și vor putea descrie pașii necesari testării produsului construit ▪ Utilizeze tehnologiile Java în implementarea componentelor funcționale și de testare ale unui proiect dat ▪ Analizeze cerințele clientului, analizeze erorile software care apar pe parcursul dezvoltării proiectului, analizeze testele manuale pe care le execută asupra unui sistem ▪ Calculeze necesarul de timp, necesarul de personal și necesarul de bani pentru dezvoltarea proiectului cerut de client

8. Conţinut

8.1	**Curs**	**Metode de predare**	**Observaţii** (ore şi referinţe bibliografice)
1.	Conținutul cursului. Bibliografie. Motivație, Definiții, Erori celebre, Statistici. Modele de dezvoltare (Cascadă, Spirală)	Prezentare de slide-uri. Note de curs şi tutoriale disponibile în versiune electronică.	2 ore, [1], [2], [3], [4], [5], [6]
2.	Modele de dezvoltare (RUP, V-Model, XP, Agile, Lean, Scrum, AMDD, TDD). Ingineria cerințelor (Actor, Use Case)	Prezentare de slide-uri. Note de curs şi tutoriale disponibile în versiune electronică.	2 ore, [1], [7], [8]
3.	Modelare, Limbaje de Modelare, UML, Diagrame Use Case, Diagrame de Clase	Prezentare de slide-uri. Note de curs şi tutoriale disponibile în versiune electronică. Desenarea la tablă a soluțiilor pentru o problemă dată.	2 ore, [1], [9]
4.	Diagrame UML (Secventa, Colaborare, Stari, Activitate,	Prezentare de slide-uri. Note de curs şi tutoriale disponibile în versiune electronică.	2 ore, [4]

	Deployment, Pachete). Metodologia SCRUM	Problematizare.	
5.	Reverse Engineering. Prezentarea uneltelor necesare dezvoltării de proiecte: Github, Trello, GoogleDocs.	Prezentare de slide-uri. Note de curs şi tutoriale disponibile în versiune electronică.	2 ore, [4], [10]
6.	GRASP. Principii SOLID	Prezentare de slide-uri. Note de curs şi tutoriale disponibile în versiune electronică. Problematizare şi lucrul pe exemple.	2 ore, [2], [4]
7.	Design Patterns: Definiții, Elemente, Clasificare, Paternuri creaționale	Prezentare de slide-uri. Note de curs şi tutoriale disponibile în versiune electronică.	2 ore, [3], [5]
8.	Recapitularea noțiunilor studiate: ingineria cerințelor, modelare, design patterns	Problematizare. Desenarea la tablă a soluțiilor pentru o problemă dată.	2 ore, [3], [5]
9.	Design Patterns: Paternuri creaționale, Paternuri structurale	Prezentare de slide-uri. Note de curs şi tutoriale disponibile în versiune electronică.	2 ore, [3], [5]
10.	Design Patterns: Paternuri comportamentale	Prezentare de slide-uri. Note de curs şi tutoriale disponibile în versiune electronică.	2 ore, [3], [5]
11.	Design Patterns: Concurență, Testare, Distribuite. Testare software	Prezentare de slide-uri. Note de curs şi tutoriale disponibile în versiune electronică.	2 ore, [3], [5]
12.	Testare software: Introducere, metode, procese Testare manuală vs Testare automată	Prezentare de slide-uri. Note de curs şi tutoriale disponibile în versiune electronică.	2 ore, [11], [12]
13.	Calitatea Programelor, Metrici, Copyright	Prezentare de slide-uri. Note de curs şi tutoriale disponibile în versiune electronică.	2 ore
14.	Conținutul cursului de la master: Swebok, SOA, QoS, BPMN, AOP, Refactorizare	Prezentare de slide-uri. Note de curs şi tutoriale disponibile în versiune electronică.	2 ore

Bibliografie

Referinţe principale:

[1] Ian Sommerville: Software Engineering, Addison Wesley, 2001
[2] Craig Larman: Applying UML and Patterns, Addisson Wesley, 2002
[3] Erich Gamma, Richard Helm, Ralph Johnson, John Vissides: Design Patterns, Elements of Reusable Object-Oriented Software, Addisson Wesley, 1998

Referinţe suplimentare:

[4] ArgoUML: http://argouml-downloads.tigris.org/
[5] Preview of Patterns in Java Volume 2: http://www.mindspring.com/~mgrand/pattern_synopses2.htm
[6] Failure rate: http://www.it-cortex.com/Stat_Failure_Rate.htm
[7] RUP in the dialogue with Scrum: http://www.ibm.com/developerworks/rational/library/feb05/krebs
[8] Agile Manifesto: http://agilemanifesto.org/
[9] Requirements analysis process: http://www.outsource2india.com/software/RequirementAnalysis.asp
[10] Chikofsky, E.J. and Cross, J.: Reverse Engineering and Design Discovery: A Taxonomy, January 1990
[11] Junit Test Example: http://www.cs.unc.edu/~weiss/COMP401/s08-27-JUnitTestExample.doc
[12] Bug Life Cycle: http://www.buzzle.com/editorials/4-6-2005-68177.asp, http://qastation.wordpress.com/2008/06/13/process-for-bug-life-cycle/

8.2	**Laborator**	**Metode de predare**	**Observaţii** (ore şi referinţe bibliografice)
1.	Recapitularea noțiunilor de programare orientată-obiect (moștenire,	Problematizare. Discuții. Schițarea la tablă a soluțiilor	2 ore, [13]

	polimorfism, incapsulare)	pentru o problemă dată.	
2.	Fișa Cerințelor, UML - Diagrame de Clase	Problematizare. Discuții. Schițarea la tablă a soluțiilor pentru o problemă dată.	2 ore, [1], [2], [4], [9]
3.	Fișa Cerințelor, UML - Diagrame Use Case, Diagrame de Clase	Problematizare. Discuții. Schițarea la tablă a soluțiilor pentru o problemă dată.	2 ore, [4], [9]
4.	Folosirea metodologiei SCRUM în rezolvarea de probleme	Problematizare. Discuții. Schițarea la tablă a soluțiilor pentru o problemă dată.	2 ore, [4]
5.	Reverse Engineering. Folosire uneltelor Github, Trello, GoogleDocs.	Problematizare. Discuții. Schițarea la tablă a soluțiilor pentru o problemă dată.	2 ore, [3], [4], [5]
6.	Design Patterns – Paternuri creaționale, JUnit Testing	Problematizare. Discuții. Schițarea la tablă a soluțiilor pentru o problemă dată.	2 ore, [2], [4], [5], [11]
7.	Lucrul la Proiect: Studiu de Risc, Ingineria Cerințelor	Problematizare. Discuții. Schițarea la tablă a soluțiilor pentru o problemă dată.	2 ore, [4], [9]
8.	Recapitularea noțiunilor studiate	Problematizare. Discuții. Schițarea la tablă a soluțiilor pentru o problemă dată.	2 ore, [3], [5]
9.	Lucrul la Proiect: Diagrame UML	Problematizare. Discuții. Schițarea la tablă a soluțiilor pentru o problemă dată.	2 ore, [4], [9]
10.	Lucrul la Proiect: Implementare, documentaţie	Problematizare. Discuții. Schițarea la tablă a soluțiilor pentru o problemă dată.	2 ore, [3], [4], [5]
11.	Lucrul la Proiect: Implementare, documentaţie	Problematizare. Discuții. Schițarea la tablă a soluțiilor pentru o problemă dată.	2 ore, [3], [4], [5]
12.	Lucrul la Proiect: Integrare	Problematizare. Discuții. Schițarea la tablă a soluțiilor pentru o problemă dată.	2 ore
13.	Lucrul la Proiect: Testare, Evaluare	Prezentare folosind video-proiectorul. Problematizare. Discuții.	2 ore, [11]
14.	Lucrul la Proiect: Evaluare	Prezentare folosind video-proiectorul. Problematizare. Discuții.	2 ore, [11]

Bibliografie:

Referinţe principale:

[1] Ian Sommerville: Software Engineering, Addison Wesley, 2001
[2] Craig Larman: Applying UML and Patterns, Addisson Wesley, 2002
[3] Erich Gamma, Richard Helm, Ralph Johnson, John Vissides: Design Patterns, Elements of Reusable Object-Oriented Software, Addisson Wesley, 1998

Referinţe suplimentare:

[4] ArgoUML: http://argouml-downloads.tigris.org/
[5] Preview of Patterns in Java Volume 2: http://www.mindspring.com/~mgrand/pattern_synopses2.htm
[6] Failure rate: http://www.it-cortex.com/Stat_Failure_Rate.htm
[7] RUP in the dialogue with Scrum: http://www.ibm.com/developerworks/rational/library/feb05/krebs
[8] Agile Manifesto: http://agilemanifesto.org/
[9] Requirements analysis process: http://www.outsource2india.com/software/RequirementAnalysis.asp
[10] Chikofsky, E.J. and Cross, J.: Reverse Engineering and Design Discovery: A Taxonomy, January 1990
[11] Junit Test Example: http://www.cs.unc.edu/~weiss/COMP401/s08-27-JUnitTestExample.doc

[12] Bug Life Cycle: http://www.buzzle.com/editorials/4-6-2005-68177.asp,
http://qastation.wordpress.com/2008/06/13/process-for-bug-life-cycle/
[13] Lucanu D.: Principiile programării orientate-obiect,
http://thor.info.uaic.ro/~dlucanu/cursuri/poo/resurse/principiiPOO.pps

9. Coroborarea conţinutului disciplinei cu aşteptările reprezentanţilor comunităţii, asociaţiilor profesionale şi angajatorilor reprezentativi din domeniul aferent programului

Companiile de IT din Iași și din țară folosesc în dezvoltarea de proiecte de dimensiuni mari noțiuni de inginerie software. Etapele necesare dezvoltării aplicațiilor de dimensiuni mari (ingineria cerințelor, modelare, implementare, integrare, testare, deployment) sunt discutate pe larg pe parcursul cursului, iar studenții învață noțiunile de bază și parcurg efectiv aceste etape. Pe parcursul verii studenții participă la sesiuni de stagii în cadrul firmelor, fiind implicați în proiecte reale, unde aplică practic pe proiecte reale noțiunile teoretice învățate la acest curs. În urma discuțiilor cu principalii angajatori, acest curs se actualizează de la an la an, adaptându-se la cerințele actuale ale pieței de IT și nu numai.

10. Evaluare

Tip activitate	**10.1 Criterii de evaluare**	**10.2 Metode de evaluare**	**10.3 Pondere în nota finală (%)**
10.4 Curs	Punctaj minim examen: 40% din punctajul maxim ce poate fi obţinut la examen	Test scris + Bonus pentru activitatea de la curs	50 %
10.5 Laborator	Punctaj minim laborator: 50% din punctajul maxim ce poate fi obţinut la laborator (minim 50 % din punctajele temelor din primele 6 laboratoare şi minim 50 % din punctajul proiectului)	Pezență + Prezentare de soluții la temele propuse săptămânal în primele 6 săptămâni + Lucrul la proiect + Bonus pentru activități suplimentare ce au legătură cu cursul de IP	50 % (din care 40 % e lucrul la proiect)
10.6 Standard minim de performanţă [raportate la competenţele definite la punctul 7. Obiectivele disciplinei]			
Studenții vor fi capabili să identifice principalele cerințe în urma discuției cu clientul și vor fi capabili să modeleze soluția pe care acesta o așteaptă. Implementarea se va reduce la construirea claselor principale și a atributelor acestora, fără implementarea metodelor de bază. Testarea proiectului o vor putea face din punct de vedere al testării automate și manuale. Punctaj minim laborator: 50% din punctajul maxim ce poate fi obţinut la laborator fără bonusuri (minim 50 % din temele de laborator şi minim 50 % proiect) Punctaj minim examen: 40% din punctajul maxim ce poate fi obţinut la examen Pentru studenţii care îndeplinesc criteriile de promovare nota finală se stabileşte împărțind punctajul obținut la maximul punctajelor fără bonus, iar rezultatul este înmulțit cu 10. Studentul care participă la examen, va primi o notă, altfel va fi considerat absent. Dacă unul din criteriile de promovare nu este îndeplinit, studentul va obţine o nota mai mică sau egală cu 4.			