FIŞA DISCIPLINEI[1]

1. Date despre program

1.1 Instituţia de învăţământ superior	Universitatea Politehnică Timișoara
1.2 Facultatea[2] / Departamentul[3]	Facultatea de Inginerie Hunedoara / Inginerie Electrică și Informatică Industrială
1.3 Catedra	—
1.4 Domeniul de studii (denumire/cod[4])	ŞTIINŢE INGINEREŞTI APLICATE / 270
1.5 Ciclul de studii	Licenta
1.6 Programul de studii (denumire/cod/calificarea)	INFORMATICĂ INDUSTRIALĂ / 50 / Inginer

2. Date despre disciplină

2.1 Denumirea disciplinei/Categoria formativă[5]			Inginerie software / DD				
2.2 Titularul activităţilor de curs			Șef lucr. dr. ing. Ghiormez Loredana				
2.3 Titularul activităţilor aplicative[6]			Șef lucr. dr. ing. Ghiormez Loredana				
2.4 Anul de studii[7]	IV	**2.5** Semestrul	II	**2.6** Tipul de evaluare	E	**2.7** Regimul disciplinei[8]	DO

3. Timp total estimat - ore pe semestru: activități didactice directe (asistate integral sau asistate parțial) și activități de pregătire individuală (neasistate) [9]

3.1 Număr de ore asistate integral/săptămână	3,5 , format din:	**3.2** ore curs	2	**3.3** ore seminar/laborator/proiect	1,5
3.1* Număr total de ore asistate integral/sem.	49 , format din:	**3.2*** ore curs	28	**3.3*** ore seminar/laborator/proiect	21
3.4 Număr de ore asistate parțial/săptămână	, format din:	**3.5** ore practică		**3.6** ore elaborare proiect de diplomă	
3.4* Număr total de ore asistate parțial/ semestru	, format din:	**3.5*** ore practică		**3.6*** ore elaborare proiect de diplomă	
3.7 Număr de ore activități neasistate/ săptămână	1,86 , format din:	ore documentare suplimentară în bibliotecă, pe platformele electronice de specialitate și pe teren			0,36
		ore studiu individual după manual, suport de curs, bibliografie și notițe			1
		ore pregătire seminarii/laboratoare, elaborare teme de casă și referate, portofolii și eseuri			0,5
3.7* Număr total de ore activități neasistate/ semestru	26 , format din:	ore documentare suplimentară în bibliotecă, pe platformele electronice de specialitate și pe teren			5
		ore studiu individual după manual, suport de curs, bibliografie și notițe			14
		ore pregătire seminarii/laboratoare, elaborare teme de casă și referate, portofolii și eseuri			7
3.8 Total ore/săptămână [10]	5,36				
3.8* Total ore/semestru	75				
3.9 Număr de credite	3				

4. **Precondiţii** (acolo unde este cazul)

4.1 de curriculum	• Programarea calculatoarelor 1, Programarea calculatoarelor 2, Proiectarea algoritmilor, Programarea calculatoarelor și limbaje de programare, Baze de date, Programare Java
4.2 de competenţe	•

[1] Formularul corespunde Fișei Disciplinei promovată prin OMECTS 5703/18.12.2011 și cerințelor Standardelor specifice ARACIS valabile începând cu 01.10.2017.
[2] Se înscrie numele facultății care gestionează programul de studiu căruia îi aparține disciplina.
[3] Se înscrie numele departamentului căruia i-a fost încredințată susținerea disciplinei și de care aparține titularul cursului.
[4] Se înscrie codul prevăzut în HG nr.140/16.03.2017 sau în HG similare actualizate anual.
[5] Disciplina se încadrează potrivit planului de învățământ în una dintre următoarele categorii formative: disciplină fundamentală (DF), disciplină de domeniu (DD), discipină de specialitate (DS) sau disciplina complementară (DC).
[6] Prin activități aplicative se înțeleg activitățile de: seminar (S) / laborator (L) / proiect (P) / practică (Pr).
[7] Anul de studii în care este prevăzută disciplina în planul de învățământ.
[8] Disciplina poate avea unul din următoarele regimuri: disciplină impusă (DI), disciplină opțională (DO) sau disciplină facultativă (Df).
[9] Numărul de ore de la rubricile 3.1*, 3.2*,…,3.8* se obțin prin înmulțirea cu 14 (săptămâni) a numărului de ore din rubricile 3.1, 3.2,…, 3.8. Informațiile din rubricile 3.1, 3.4 și 3.7 sunt chei de verificare folosite de ARACIS sub forma: (3.1)+(3.4) ≥ 28 ore/săpt. și (3.8) ≤ 40 ore/săpt.
[10] Numărul total de ore / săptămână se obține prin însumarea numărului de ore de la punctele 3.1, 3.4 și 3.7.

5. Condiţii (acolo unde este cazul)

5.1 de desfăşurare a cursului	• Sală de curs echipată cu videoproiector și conexiune la Internet. • Studenții nu se vor prezenta la prelegeri cu telefoanele mobile deschise.
5.2 de desfăşurare a activităţilor practice	• Sală de laborator echipată cu cu videoproiector și computere. • Studenții nu se vor prezenta la activitățile practice cu telefoanele mobile deschise.

6. Competenţe la formarea cărora contribuie disciplina

Competenţe specifice	C2.1 Descrierea structurii şi a modului de funcţionare a sistemelor informatice în general; C2.2 Explicarea rolului, funcţionalităţii şi utilităţii sistemelor informatice în general şi a sistemelor de prelucrare şi gestiune a datelor în domeniul specializării. C2.3 Utilizarea componentelor software ale sistemelor informatice, folosind algoritimi, protocoale, limbaje, structuri de date; C2.4 Aprecierea caracteristicilor şi calităţii sistemelor informatice. C2.5 Prelucrarea şi gestionarea datelor utilizând sisteme informatice dedicate. C4.1 Descrierea arhitecturilor de bază pentru sistemele informatice aplicate în conducerea sistemelor energetice sau industriale. C4.2 Explicarea şi interpretarea funcţionării elementelor sistemelor informatice aferente conducerii proceselor energetice sau industriale; C4.3 Alegerea elementelor unui sistem informatic destinat conducerii, comenzii, reglajului sau supravegherii unui proces energetic sau industrial; C4.4 Utilizarea criteriilor şi metodelor de evaluare a performanţelor tehice şi informatice ale unui sistem informatic de proces; C4.5 Implementarea unei structuri de sistem informatic de conducere a proceselor din sistemele energetice sau industriale. C5.1 Descrierea structurilor de conducere automată bazate pe microprocesoare şi microcontrolere; C5.2 Explicarea utilizării microprocesoarelor şi microcontrolerelor şi cunoaşterea softului aferent acestora; C5.3 Modelarea, simularea şi testarea sistemelor de conducere automată a proceselor industriale; C5.4 Evaluarea performanţelor de regim staţionar şi dinamic ale sistemelor de conducere automată; • C5.5 Realizarea unui sistem de comandă şi reglare automată a unui proces industrial specific domeniului specializării.
Competențele profesionale în care se înscriu competențele specifice	C2 Utilizarea sistemelor informatice de prelucrare şi gestiune a datelor. C4 Realizarea şi implementarea sistemelor informatice de conducere, comandă, reglaj şi supraveghere a proceselor energetice sau industrial • C5 Analiza şi sinteza sistemelor de conducere a proceselor industriale bazate pe microprocesoare şi microcontrolere.

Competențele transversale în care se înscriu competențele specifice	•

7. Obiectivele disciplinei (asociate competențelor de la punctul 6)

7.1 Obiectivul general al disciplinei	• Scopul acestei discipline îl reprezintă însuşirea principalelor concepte, metode şi tehnici ale ingineriei software, cu accent pe reutilizarea codului sursă atât din punct de vedere al programării orientate pe obiect, cât şi din punct de vedere al programării generice în limbajul de programare Java.
7.2 Obiectivele specifice	Obiectivele specifice ale acestei discipline sunt: • Cunoaşterea etapelor ciclului de viaţă al unui sistem soft; • Înţelegerea conceptelor legate de modelarea softului; • Familiarizarea cu limbajul de modelare UML; • Abilitatea de a utiliza instrumente CASE.

8. Conţinuturi[11]

8.1 Curs	Număr de ore	Metode de predare[12]
1. Introducere în ingineria software 1.1. Context și motivație 1.2. Introducere 1.3. Fazele procesului de dezvoltare 1.4. Istoria programării 1.5. Concluzii	2	Studenții au acces la curs în format electronic. Se vor utiliza atât prezentări interactive cât și tradiționale. Se vor folosi: problematizarea, studiu de caz, conversația.
2. Limbajul unificat de modelare UML 2.1. Conceptul de modelare 2.2. Scurt istoric 2.3. Clasificarea diagramelor UML 2.4. Diagrame specifice fazei de analiză 2.5. Diagrame specifice fazei de proiectare 2.6. Diagrame specifice fazei de implementare 2.7. Instrumente CASE: IBM Rational Rhapsody, Microsoft Visio, ArgoUML, StarUML Exemple implementate în limbajul de programare Java	4	
3. Metodologii de dezvoltare a programelor 3.1. Ciclul de viață al unui produs software 3.2. Tipuri de activități 3.3. Metedologii generice 3.3.1. Metodologia secvențială 3.3.2. Metodologia ciclică 3.3.3. Metodologia hibridă 3.4. Metedologii concrete 3.4.1. Metodologia cascadă 3.4.2. Metodologia spirală 3.4.3. Prototipizare 3.4.4. Modelul V 3.4.5. Metode formale 3.4.6. Metodologia Agile 3.4.6.1. Programare extremă 3.4.6.2. SCRUM	4	
4. Faza de analiză a procesului de dezvoltare 4.1. Introducere 4.2. Metode de analiză	2	

[11] Se detaliază toate activitățile didactice prevăzute prin planul de învățământ (tematicile prelegerilor și ale seminariilor, lista lucrărilor de laborator, conținuturile etapelor de elaborare a proiectelor, tematica fiecărui stagiu de practică). Titlurile lucrărilor de laborator care se efectuează pe standuri vor fi însoțite de notația „(*)".
[12] Prezentarea metodelor de predare va include și folosirea noilor tehnologii (e-mail, pagină personalizată de web, resurse în format electronic etc.).

4.3. Tipuri de specificații 4.4. Documentul specificațiilor cerințelor software 4.5. Validarea software-ului 4.6. Concluzii		
5. Faza de proiectare a procesului de dezvoltare 5.1. Introducere 5.2. Arhitectura software 5.3. Proiectarea modulelor 5.4. Principii de proiectare. Exemple implementate în limbajul de programare Java 5.5. Validarea software-ului 5.6. Șabloane de proiectare 5.7. Concluzii	2	
6. Şabloane de proiectare creaționale 6.1. Fabrica simplă 6.2. Metoda fabrică 6.3. Fabrica abstractă 6.4. Singleton 6.5. Prototip 6.6. Constructor 6.4. Concluzii Exemple implementate în limbajul de programare Java	4	
7. Şabloane de proiectare arhitecturale 7.1. Șablonul arhitectural MVP 7.2. Șablonul arhitectural MVC Exemple implementate în limbajul de programare Java, respectiv reprezentări de diagrame UML, aferente problemelor rezolvate	4	
8. Managementul unui proiect software 8.1. Managementul software 8.2. Managementul configuraţiei 8.3. Managementul echipei 8.4. Estimarea costului unui sistem	2	
9. Verificarea şi validarea sistemelor soft 9.1. Testarea programelor 9.2. Instrumente de testare: JUnit 9.3. Verificarea modelelor, Verificarea specificaţiei şi a corectitudinii algoritmilor 9.4. Testarea modulelor şi a sistemului final 9.5. Verificarea documentaţiei 9.6. Validarea programelor 9.7. Certificarea produselor soft	4	

Bibliografie[13]

1. Ghiormez L., Curs în format electronic - Inginerie software, Campusul virtual al UPT, https://cv.upt.ro/course/view.php?id=2716
2.Dorin Bocu, Răzvan Bocu, Provocări și metode de abordare în managementul proiectelor IT, Editura Albastră, Cluj-Napoca,2013
3.Dorin Bocu, Răzvan Bocu, Modelare obiect orientată cu UML, Editura Albastră, Cluj-Napoca, 2006
4.Mircea Cezar Preda, Ana-Maria Mirea, Doina Lavinia Preda, Constantin Teodorescu-Mihai, Introducere în programareaorientată-obiect. Concepte fundamentale din perspectiva ingineriei software, Editura Polirom, Iași, 2010
5.Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Șabloane de proiectare. Elemente de software reutilizabilorientat pe obiect, Editura Teora, 2002
6 Militon Frențiu, Verificarea și validarea sistemelor soft, Presa Universitară Clujană, Cluj-Napoca, 2010
7.Philip Wadler, Maurice Naftali, Java Generics and Collections, O'Reilly Media, 2013
8.Pankaj Jalote, A Concise Introduction to Software Engineering, Springer-Verlag Londra, 2008
9.Philip Wadler, Maurice Naftali, Java Generics and Collections, O'Reilly Media, 2013
10.Tanasa S., Olaru C., Java de la 0 la expert, Editura Polirom, Colectia Calculatoare, 2011.

[13] Cel puțin un un titlu trebuie să aparțină colectivului disciplinei iar cel puțin un titlu trebuie să se refere la o lucrare de referință pentru disciplină, de circulație națională și internațională, existentă în biblioteca UPT.

11. *Head First Design Patterns*, O'Reilly Media, Inc. ● http://www.msquaredweb.com/DesignPatterns/ HeadFirstDesignPatternsInCSharp.zip 12. Data & Object Factory. *Design Patterns* ● http://www.dofactory.com/Patterns/Patterns.aspx		
8.2 Activităţi aplicative[14]	Număr de ore	Metode de predare
1. Elaborarea diagramelor specifice fazei de analiză: diagrama cazurilor de utilizare utilizând StarUML	4	Se va utiliza exercițiul la tablă și implementarea programului utilizând calculatorul.
2. Elaborarea diagramelor specifice fazei de analiză: diagrame de activităţi utilizând StarUML	4	
3. Elaborarea diagramelor specifice fazei de proiectare: diagrama de clase, diagrama de pachete utilizând StarUML	4	
4. Elaborarea diagramelor specifice fazei de proiectare: diagrame de stare utilizând StarUML	2	
5. Elaborarea diagramelor specifice fazei de proiectare: diagrame de secvenţă, diagrame de colaborare utilizând StarUML	2	Se va utiliza exercițiul la tablă și implementarea programului utilizând calculatorul.
6. Elaborarea diagramelor specifice fazei de implementare: diagrama de componente utilizând StarUML	2	Se va utiliza exercițiul la tablă și implementarea programului utilizând calculatorul.
7. Analiza, proiectarea şi implementarea de aplicaţii în limbajul Java ce utilizează şablonul arhitectural MVC. Testarea aplicațiilor folosind mediul NetBeans IDE. Recuperări de laborator.	3	Se va utiliza exercițiul la tablă și implementarea programului utilizând calculatorul.
Bibliografie[15] 1. Ghiormez L., Laborator în format electronic - Inginerie software, Campusul virtual al UPT, https://cv.upt.ro/course/view.php?id=2716 2. Anca Daniela Ioniță, Modelarea UML în ingineria sistemelor de programare, Editura All, București, 2003 3. Dorin Bocu, Iniţiere în ingineria sistemelor soft, Editura Albastră, Cluj-Napoca, 2002 4. Militon Frențiu, Verificarea și validarea sistemelor soft, Presa Universitară Clujană, Cluj-Napoca, 2010 5. Philip Wadler, Maurice Naftali, Java Generics and Collections, O'Reilly Media, 2013 6. John Hunt, Guide to the Unified Process Featuring UML, Java and Design Patterns, Springer London Ltd, 2014 7. Hamill Paul, Unit Test Frameworks, O'Reilly Media, 2016 8. Tanasa S., Olaru C., Java de la 0 la expert, Editura Polirom, Colectia Calculatoare, 2011.		

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

- Disciplina vine în întâmpinarea așteptărilor angajatorilor reprezentativi din domeniul aferent programului prin conținutul orelor de curs și laborator.

10. Evaluare

Tip activitate	**10.1** Criterii de evaluare[16]	**10.2** Metode de evaluare	**10.3** Pondere din nota finală
10.4 Curs	Cunoștințe teoretice	Scris - subiecte teoretice și aplicații sau proiect individual sau în echipă de două persoane. Tema proiectului se distribuie aleatoriu. Rezolvările în cazului subiectului scris se încarcă pe campusul virtual UPT.	0,66

[14] Tipurile de activități aplicative sunt cele precizate în nota de subsol 5. Dacă disciplina conține mai multe tipuri de activități aplicative atunci ele se trec consecutiv în liniile tabelului de mai jos. Tipul activității se va înscrie într-o linie distinctă sub forma: „Seminar:", „Laborator:", „Proiect:" și/sau „Practică:".
[15] Cel puțin un titlu trebuie să aparțină colectivului disciplinei.
[16] Fișele disciplinelor trebuie să conțină procedura de evaluare a disciplinei cu precizarea criteriilor, a metodelor și a formelor de evaluare, precum și cu precizarea ponderilor atribuite acestora în nota finală. Criteriile de evaluare se formulează în mod distinct pentru fiecare activitate prevăzută în planul de învățământ (curs, seminar, laborator, proiect). Ele se vor referi și la formele de verificare pe parcurs (teme de casă, referate ș.a.)

		Examenul se desfășoară prin intermediul Campusului Virtual UPT sau se primesc subiecte tipărite. În caz de proiect, acesta va fi prezentat oral și va exista legătura între diagramele UML, respectiv aplicația scrisă în limbaj de programare. În caz de proiect în echipă, un student va prezenta diagramele UML, iar celălalt student va prezenta aplicația. În caz de scenariu online se utilizează și aplicația de videoconferință (Zoom)	
10.5 Activităţi aplicative	**S:**		
	L: Abilități în analiza, proiectarea și implementarea aplicațiilor de laborator	Oral – aplicații utilizând calculatorul In caz de scenariu online testele de control se desfășoară prin intermediul campusului virtual UPT	0,34
	P[17]**:**		
	Pr:		

10.6 Standard minim de performanţă (se prezintă cunoștințele minim necesare pentru promovarea disciplinei și modul în care se verifică stăpânirea lor[18])

- Nota 5 la activitățile aplicative constă în implementarea a cel puțin 4 diagrame UML implementate pe aceeași tematică, dintre care una obligatoriu să fie diagrama de clase.
- Nota 5 la examenul final se obține în condițiile obținerii a minim jumătate din punctajul total alocat subiectelor.
- Pentru a promova disciplina trebuie ca ambele activități (curs și laborator) să fie promovate.
-

Data completării

05.10.2023

Director de departament
(semnătura)

.....

Titular de curs
(semnătura)

.....

Data avizării în Consiliul Facultăţii[19]

16.10.2023

Titular activităţi aplicative
(semnătura)

.....

Decan
(semnătura)

.....

[17] În cazul când proiectul nu este o disciplină distinctă, în această rubrică se va preciza și modul în care rezultatul evaluării proiectului condiționează admiterea studentului la evaluarea finală din cadrul disciplinei.
[18] Nu se va explica cum se acorda nota de promovare.
[19] Avizarea este precedată de discutarea punctului de vedere al board-ului de care aparține programul de studii cu privire la fișa disciplinei.