FIŞA DISCIPLINEI

1. Date despre program

1.1 Instituţia de învăţământ superior	**Universitatea Babeş-Bolyai Cluj-Napoca**
1.2 Facultatea	**Facultatea de Matematică şi Informatică**
1.3 Departamentul	**Departamentul de informatică**
1.4 Domeniul de studii	**Informatică**
1.5 Ciclul de studii	**Master**
1.6 Programul de studiu / Calificarea	**Informatică didactică - în limba română**

2. Date despre disciplină

2.1 Denumirea disciplinei	**Programare orientată obiect (pentru perfecționarea profesorilor)**						
2.2 Titularul activităţilor de curs	**Prof. dr. CZIBULA István Gergely**						
2.3 Titularul activităţilor de seminar	**Prof. dr. CZIBULA István Gergely**						
2.4 Anul de studiu	**1**	2.5 Semestrul	**1**	2.6. Tipul de evaluare	**E**	2.7 Regimul disciplinei	**Obligatorie**

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

3.1 Număr de ore pe săptămână	4	Din care: 3.2 curs	2	3.3 seminar/laborator	1 sem +1pr
3.4 Total ore din planul de învăţământ	56	Din care: 3.5 curs	28	3.6 seminar/laborator	28
Distribuţia fondului de timp:					ore
Studiul după manual, suport de curs, bibliografie şi notiţe					28
Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate şi pe teren					14
Pregătire seminarii/laboratoare, teme, referate, portofolii şi eseuri					30
Tutoriat					12
Examinări					10
Alte activităţi:					

3.7 Total ore studiu individual	94
3.8 Total ore pe semestru	150
3.9 Numărul de credite	6

4. Precondiţii (acolo unde este cazul)

4.1 de curriculum	• Algoritmică și programare • Structuri de date și complexități
4.2 de competenţe	Abilităţi medii de programare

5. Condiţii (acolo unde este cazul)

5.1 De desfăşurare a cursului	• Sală de curs cu videoproiector
5.2 De desfăşurare a seminarului/laboratorului	

6. Competenţele specifice acumulate

Competenţe profesionale	• Cunoaşterea şi înţelegerea conceptelor specifice programării orientate obiect • Înțelegerea rolului moștenirii, polimorfismului, legării dinamice și structurilor dinamice în realizarea de cod reutilizabil. • Abilități de programare modulară și orientată obiect în limbajele de programare C++ și Python. • Însuşirea celor mai bune recomandări privind dezvoltarea aplicaţiilor cu o arhitectură stratificată minimală. • Înţelegerea testării şi verificării în scrierea unor programe de calitate.
Competenţe transversale	• Abilitatea de a aplica conceptele, principiilor şi tehnicilor însuşite în rezolvarea unor probleme reale. • Capacitate de muncă independentă • Manifestarea unei atitudini responsabile in activitatile didactice si stiintifice. • Respectarea principiilor profesionale si etice.

7. Obiectivele disciplinei (reieşind din grila competenţelor acumulate)

7.1 Obiectivul general al disciplinei	Să deprindă studentul cu proiectare orientată obiect a problemelor de scară mică/mijlocie, aprofundarea limbajului de programare Python și învățarea limbajului de programare C++.
7.2 Obiectivele specifice	• Demonstrarea diferenței dintre proiectarea imperativă tradițională și proiectarea oriantată obiect. • Înțelegerea rolului moștenirii, polimorfismului, legării dinamice și a structurilor generice în realizarea codului reutilizabil. • Explicarea și utilizarea dezvoltării bazate pe funcționalități, dezvoltarea bazată pe testare, utilizarea aserțiunilor formale și tratarea excepțiilor. • Scrierea de programme de scară mică/mijlocie folosind limbajele C++ și Python. • Utilizarea claselor scrise de alți programatori în dezvoltarea sistemelor. • Însușirea unui stil de programare conform celor mai bune recomandari practice

8. Conţinuturi

8.1 Curs	Metode de predare	Observaţii
1. Introducere in procese de dezvoltare software • Cum scriem programe: enunt problema, cerinte, proces de dezvoltare dirijat de functionalitati (FDD) • Programare dirijata de teste, refactorizari	• Expunerea interactivă • Explicaţia • Conversaţia • Demonstraţia didactică • Exemple	
2. Programare modulara • Cum organizam codul sursa: responsabilitati, single responsibility principle, separation of concerns, dependency, coupling, cohesion • Arhitecturi software stratificate	• Expunerea interactivă • Explicaţia • Conversaţia • Demonstraţia didactică • Exemple	
3-4. Programare orientată obiect în C++ și Python. • Clase și obiecte. • Membri unei clase. Modificatori de acces.	• Expunerea interactivă • Explicaţia • Conversaţia	

• Constructori/destructori. Diagrame UML pentru clase (membri, acces).	• Demonstraţia didactică • Exemple	
5. Programare prin abstractizarea datelor. • Tipuri abstracte de date. • Încapsularea și ascunderea informației. • Avantaje ale utilizării tipurilor abstracte de date în proiectarea aplicațiilor.	• Expunerea interactivă • Explicaţia • Conversaţia • Demonstraţia didactică • Exemple	
6-7. Principii de proiectare si programare • Problema: program cu operatii CRUD pe entitati de un tip dat • Arhitectura stratificata: UI, Domeniu, Infrastructura • Sabloane GRASP • Sabloane DDD: entity, validator, repository, controller • Principii: Information Expert, Low Coupling, High Cohesion, Protected Variation, Single responsibility, Dependency Injection	• Expunerea interactivă • Explicaţia • Conversaţia • Demonstraţia didactică • Exemple	
8-9. Moștenire • Moștenire simplă. Clase derivate. • Principiul substituției. • Supraîncărcarea metodelor. • Moștenire multiplă. • Relații de specializare/generalizare – reprezentări UML.	• Expunerea interactivă • Explicaţia • Conversaţia • Demonstraţia didactică • Exemple	
10. Operații de intrare/ieșire în C++ și Python. Tratarea excepțiilor • Stream-uri I/O. Ierarhia de clase I/O. • Formatare. Manipulatori. • Fișiere text.	• Expunerea interactivă • Explicaţia • Conversaţia • Demonstraţia didactică • Exemple	
11. Biblioteca STL • Iteratori STL. • Algoritmi STL.	• Expunerea interactivă • Explicaţia • Exemple	
12-14. Prezentări referate	• Expunerea interactivă • Conversaţia	

Bibliografie

1. B. Stroustup, The C++ Programming Language, Addison Wesley, 1998.
2. Bruce Eckel, Thinking in C++, www.bruceeckel.com
3. Alexandrescu, Programarea moderna in C++. Programare generica si modele de proiectare aplicate, Editura Teora, 2002
4. M. Frentiu, B. Parv, Elaborarea programelor. Metode si tehnici moderne, Ed. Promedia, Cluj-Napoca, 1994.
5. E. Horowitz, S. Sahni, D. Mehta, Fundamentals of Data Structures in C++, Computer Science Press, Oxford, 1995.
6. K.A. Lambert, D.W. Nance, T.L. Naps, Introduction to Computer Science with C++, West Publishing Co., New-York, 1996.
7. L. Negrescu, Limbajul C++, Ed. Albastra,Cluj-Napoca 1996.

8.2 Seminar	Metode de predare	Observaţii
1. Aspecte administrative	• Conversația • Dezbaterea	Seminarul este structurat sub forma a 2 ore din 2 în 2 săptămâni
2. Stabilirea temei pentru proiectul software	• Conversația • Dezbaterea • Studii de caz • Exemple	
3. Prezentări de referate și rapoarte privind derularea	• Conversația	

proiectelor	• Dezbaterea • Studii de caz • Prezentări	
4. Prezentări de referate & rapoarte privind derularea proiectelor	• Conversația • Dezbaterea • Studii de caz • Prezentări	
5. Prezentări de referate și rapoarte privind derularea proiectelor	• Conversația • Dezbaterea • Studii de caz • Prezentări	
6. Prezentări de referate și rapoarte privind derularea proiectelor	• Conversația • Dezbaterea • Studii de caz • Prezentări	
7. Demo cu aplicația realizată la proiectul software	Expuneri, demo-uri	

Bibliografie

Studenții vor căuta și folosi

- documentații referitoare la paradigma de programare orientată obiect
- articole de cercetare în principalele baze de date cu lucrări de computer science

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

- Acest curs urmează recomandările IEEE și ACM pentru studii în programarea orientată obiect.
- Conţinutul disciplinei este în concordanţă cu ceea ce se face în alte centre universitare din tară.
- Conţinutul disciplinei este important pentru a îmbunătăți deprinderile de dezvoltare de software.

10. Evaluare

Tip activitate	10.1 Criterii de evaluare	10.2 Metode de evaluare	10.3 Pondere din nota finală(%)
10.4 Curs	• cunoașterea conceptelor de bază ale programării orientate obiect • aplicarea paradigmei OOP la diverse domenii de problemă/aplicație	Examen scris	40%
10.5 Seminar/ activități laborator	• capacitatea de a investiga și studia literatura referitoare la paradigma OOP • capacitatea de a rezolva o problemă folosind paradigma OOP	• Referat teoretic • Proiect software • Prezența la seminar/laborator • Oficiu	20% 20% 10% 10%
10.6 Standard minim de performanţă			
• Cel puțin nota 5 (pe o scară de la 1 la 10) la examenul scris, referat și proiect.			

Data completării	Semnătura titularului de curs	Semnătura titularului de seminar
10.04.2018	Prof. dr. Czibula István Gergely	Prof. dr. Czibula István Gergely

Data avizării în departament — Semnătura directorului de departament

Prof. dr. Anca Andreica