

## FIȘA DISCIPLINEI

### 1. Date despre program

1.1 Instituția de învățământ superior	<b>Universitatea Babeș-Bolyai Cluj-Napoca</b>
1.2 Facultatea	<b>Facultatea de Matematică și Informatică</b>
1.3 Departamentul	<b>Departamentul de informatică</b>
1.4 Domeniul de studii	<b>Informatică</b>
1.5 Ciclul de studii	<b>Master</b>
1.6 Programul de studiu / Calificarea	<b>Informatică didactică - în limba română</b>

### 2. Date despre disciplină

2.1 Denumirea disciplinei	<b>Programare orientată obiect (pentru perfecționarea profesorilor)</b>						
2.2 Titularul activităților de curs	<b>Conf. dr. CZIBULA István Gergely</b>						
2.3 Titularul activităților de seminar	<b>Conf. dr. CZIBULA István Gergely</b>						
2.4 Anul de studiu	<b>1</b>	2.5 Semestrul	<b>2</b>	2.6. Tipul de evaluare	<b>E</b>	2.7 Regimul disciplinei	<b>Obligatorie</b>

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

3.1 Număr de ore pe săptămână	3	Din care: 3.2 curs	2	3.3 seminar/laborator	1
3.4 Total ore din planul de învățământ	42	Din care: 3.5 curs	28	3.6 seminar/laborator	14
Distribuția fondului de timp:					ore
Studiul după manual, suport de curs, bibliografie și notițe					36
Documentare suplimentară în bibliotecă, pe platformele electronice de specialitate și pe teren					25
Pregătire seminarii/laboratoare, teme, referate, portofolii și eseuri					44
Tutoriat					18
Examinări					10
Alte activități: .....					
3.7 Total ore studiu individual			133		
3.8 Total ore pe semestru			175		
3.9 Numărul de credite			7		

### 4. Precondiții (acolo unde este cazul)

4.1 de curriculum	<ul style="list-style-type: none"> <li>• Algoritmă și programare</li> <li>• Structuri de date și complexități</li> </ul>
4.2 de competențe	Abilități medii de programare

### 5. Condiții (acolo unde este cazul)

5.1 De desfășurare a cursului	<ul style="list-style-type: none"> <li>• Sală de curs cu videoproiector</li> </ul>
5.2 De desfășurare a seminarului/laboratorului	

### 6. Competențele specifice acumulate

<b>Competențe profesionale</b>	<ul style="list-style-type: none"> <li>• Cunoașterea și înțelegerea conceptelor specifice programării orientate obiect</li> <li>• Înțelegerea rolului moștenirii, polimorfismului, legării dinamice și structurilor dinamice în realizarea de cod reutilizabil.</li> <li>• Abilități de programare modulară și orientată obiect în limbajele de programare C++ și Python.</li> <li>• Înșușirea celor mai bune recomandări privind dezvoltarea aplicațiilor cu o arhitectură stratificată minimală.</li> <li>• Înțelegerea testării și verificării în scrierea unor programe de calitate.</li> </ul>
<b>Competențe transversale</b>	<ul style="list-style-type: none"> <li>• Abilitatea de a aplica conceptele, principiilor și tehnicilor însușite în rezolvarea unor probleme reale.</li> <li>• Capacitate de muncă independentă</li> <li>• Manifestarea unei atitudini responsabile în activitățile didactice și științifice.</li> <li>• Respectarea principiilor profesionale și etice.</li> </ul>

## 7. Obiectivele disciplinei (reieșind din grila competențelor 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	<ul style="list-style-type: none"> <li>• Demonstrarea diferenței dintre proiectarea imperativă tradițională și proiectarea orientată obiect.</li> <li>• Înțelegerea rolului moștenirii, polimorfismului, legării dinamice și a structurilor generice în realizarea codului reutilizabil.</li> <li>• Explicarea și utilizarea dezvoltării bazate pe funcționalități, dezvoltarea bazată pe testare, utilizarea aserțiunilor formale și tratarea excepțiilor.</li> <li>• Scrierea de programe de scară mică/mijlocie folosind limbajele C++ și Python.</li> <li>• Utilizarea claselor scrise de alți programatori în dezvoltarea sistemelor.</li> <li>• Înșușirea unui stil de programare conform celor mai bune recomandări practice</li> </ul>

## 8. Conținuturi

8.1 Curs	Metode de predare	Observații
<b>1. Introducere în procese de dezvoltare software</b> <ul style="list-style-type: none"> <li>• Cum scriem programe: enunț problema, cerințe, proces de dezvoltare dirijat de funcționalități (FDD)</li> <li>• Programare dirijată de teste, refactorizări</li> </ul>	<ul style="list-style-type: none"> <li>• Expunerea interactivă</li> <li>• Explicația</li> <li>• Conversația</li> <li>• Demonstrația didactică</li> <li>• Exemple</li> </ul>	
<b>2. Programare modulară</b> <ul style="list-style-type: none"> <li>• Cum organizăm codul sursă: responsabilități, single responsibility principle, separation of concerns, dependency, coupling, cohesion</li> <li>• Arhitecturi software stratificate</li> </ul>	<ul style="list-style-type: none"> <li>• Expunerea interactivă</li> <li>• Explicația</li> <li>• Conversația</li> <li>• Demonstrația didactică</li> <li>• Exemple</li> </ul>	
<b>3-4. Programare orientată obiect în C++ și Python.</b> <ul style="list-style-type: none"> <li>• Clase și obiecte.</li> <li>• Membri unei clase. Modificatori de acces.</li> </ul>	<ul style="list-style-type: none"> <li>• Expunerea interactivă</li> <li>• Explicația</li> <li>• Conversația</li> </ul>	

<ul style="list-style-type: none"> <li>• Constructori/destructori.</li> </ul> Diagrame UML pentru clase (membri, acces).	<ul style="list-style-type: none"> <li>• Demonstrația didactică</li> <li>• Exemple</li> </ul>	
<b>5. Programare prin abstractizarea datelor.</b> <ul style="list-style-type: none"> <li>• Tipuri abstracte de date.</li> <li>• Încapsularea și ascunderea informației.</li> <li>• Avantaje ale utilizării tipurilor abstracte de date în proiectarea aplicațiilor.</li> </ul>	<ul style="list-style-type: none"> <li>• Expunerea interactivă</li> <li>• Explicația</li> <li>• Conversația</li> <li>• Demonstrația didactică</li> <li>• Exemple</li> </ul>	
<b>6-7. Principii de proiectare și programare</b> <ul style="list-style-type: none"> <li>• Problema: program cu operații CRUD pe entități de un tip dat</li> <li>• Arhitectura stratificată: UI, Domeniu, Infrastructura</li> <li>• Sabloane GRASP</li> <li>• Sabloane DDD: entity, validator, repository, controller</li> <li>• Principii: Information Expert, Low Coupling, High Cohesion, Protected Variation, Single responsibility, Dependency Injection</li> </ul>	<ul style="list-style-type: none"> <li>• Expunerea interactivă</li> <li>• Explicația</li> <li>• Conversația</li> <li>• Demonstrația didactică</li> <li>• Exemple</li> </ul>	
<b>8-9. Moștenire</b> <ul style="list-style-type: none"> <li>• Moștenire simplă. Clase derivate.</li> <li>• Principiul substituției.</li> <li>• Supraîncărcarea metodelor.</li> <li>• Moștenire multiplă.</li> <li>• Relații de specializare/generalizare – reprezentări UML.</li> </ul>	<ul style="list-style-type: none"> <li>• Expunerea interactivă</li> <li>• Explicația</li> <li>• Conversația</li> <li>• Demonstrația didactică</li> <li>• Exemple</li> </ul>	
<b>10. Operații de intrare/ieșire în C++ și Python. Tratarea excepțiilor</b> <ul style="list-style-type: none"> <li>• Stream-uri I/O. Ierarhia de clase I/O.</li> <li>• Formatare. Manipulatori.</li> <li>• Fișiere text.</li> </ul>	<ul style="list-style-type: none"> <li>• Expunerea interactivă</li> <li>• Explicația</li> <li>• Conversația</li> <li>• Demonstrația didactică</li> <li>• Exemple</li> </ul>	
<b>11. Biblioteca STL</b> <ul style="list-style-type: none"> <li>• Iteratori STL.</li> <li>• Algoritmi STL.</li> </ul>	<ul style="list-style-type: none"> <li>• Expunerea interactivă</li> <li>• Explicația</li> <li>• Exemple</li> </ul>	
<b>12-14. Prezentări referate</b>	<ul style="list-style-type: none"> <li>• Expunerea interactivă</li> <li>• Conversația</li> </ul>	
<b>Bibliografie</b> <ol style="list-style-type: none"> <li>1. B. Stroustup, The C++ Programming Language, Addison Wesley, 1998.</li> <li>2. Bruce Eckel, Thinking in C++, www.bruceeckel.com</li> <li>3. Alexandrescu, Programarea modernă în C++. Programare generică și modele de proiectare aplicate, Editura Teora, 2002</li> <li>4. M. Frentiu, B. Parv, Elaborarea programelor. Metode și tehnici moderne, Ed. Promedia, Cluj-Napoca, 1994.</li> <li>5. E. Horowitz, S. Sahni, D. Mehta, Fundamentals of Data Structures in C++, Computer Science Press, Oxford, 1995.</li> <li>6. K.A. Lambert, D.W. Nance, T.L. Naps, Introduction to Computer Science with C++, West Publishing Co., New-York, 1996.</li> <li>7. L. Negrescu, Limbajul C++, Ed. Alabastra, Cluj-Napoca 1996.</li> </ol>		
<b>8.2 Seminar</b>	Metode de predare	Observații
1. Aspecte administrative	<ul style="list-style-type: none"> <li>• Conversația</li> <li>• Dezbateră</li> </ul>	Seminarul este structurat sub forma a 2 ore din 2 în 2 săptămâni
2. Stabilirea temei pentru proiectul software	<ul style="list-style-type: none"> <li>• Conversația</li> <li>• Dezbateră</li> <li>• Studii de caz</li> <li>• Exemple</li> </ul>	
3. Prezentări de referate și rapoarte privind derularea	<ul style="list-style-type: none"> <li>• Conversația</li> </ul>	

proiectelor	<ul style="list-style-type: none"> <li>• Dezbateră</li> <li>• Studii de caz</li> <li>• Prezentări</li> </ul>	
4. Prezentări de referate & rapoarte privind derularea proiectelor	<ul style="list-style-type: none"> <li>• Conversația</li> <li>• Dezbateră</li> <li>• Studii de caz</li> <li>• Prezentări</li> </ul>	
5. Prezentări de referate și rapoarte privind derularea proiectelor	<ul style="list-style-type: none"> <li>• Conversația</li> <li>• Dezbateră</li> <li>• Studii de caz</li> <li>• Prezentări</li> </ul>	
6. Prezentări de referate și rapoarte privind derularea proiectelor	<ul style="list-style-type: none"> <li>• Conversația</li> <li>• Dezbateră</li> <li>• Studii de caz</li> <li>• Prezentări</li> </ul>	
7. Demo cu aplicația realizată la proiectul software	Expuneri, demo-uri	
<b>Bibliografie</b> Studenții vor căuta și folosi <ul style="list-style-type: none"> <li>• documentații referitoare la paradigma de programare orientată obiect</li> <li>• articole de cercetare în principalele baze de date cu lucrări de computer science</li> </ul>		

### 9. Coroborarea conținuturilor disciplinei cu așteptările reprezentanților comunității epistemice, asociațiilor profesionale și angajatori reprezentativi din domeniul aferent programului

- Acest curs urmează recomandările IEEE și ACM pentru studii în programarea orientată obiect.
- Conținutul disciplinei este în concordanță cu ceea ce se face în alte centre universitare din țară.
- Conținutul 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	<ul style="list-style-type: none"> <li>• cunoașterea conceptelor de bază ale programării orientate obiect</li> <li>• aplicarea paradigmei OOP la diverse domenii de problemă/aplicație</li> </ul>	Examen scris	40%
10.5 Seminar/activități laborator	<ul style="list-style-type: none"> <li>• capacitatea de a investiga și studia literatura referitoare la paradigma OOP</li> <li>• capacitatea de a rezolva o problemă folosind paradigma OOP</li> </ul>	<ul style="list-style-type: none"> <li>• Referat teoretic</li> <li>• Proiect software</li> <li>• Prezența la seminar/laborator</li> <li>• Oficiu</li> </ul>	20% 20% 10% 10%
10.6 Standard minim de performanță			
<ul style="list-style-type: none"> <li>• Cel puțin nota 5 (pe o scară de la 1 la 10) la examenul scris, referat și proiect.</li> </ul>			

Data completării

30.03.2016

Data avizării în departament

Semnătura titularului de curs

Conf. dr. Czibula István Gergely

Semnătura titularului de seminar

Conf. dr. Czibula István Gergely

Semnătura directorului de departament

Prof. dr. Anca Andreica