FIŞA DISCIPLINEI

1. Date despre program

1.1 Instituţia de învăţământ superior	Universitatea Politehnica Timișoara
1.2 Facultatea[1] / Departamentul[2]	Inginerie Hunedoara/Inginerie Electrică și Informatică Industrială
1.3 Domeniul de studii (denumire/cod [3])	Inginerie electrică / 90
1.4 Ciclul de studii	Licență
1.5 Programul de studii (denumire/cod/calificarea)	Inginerie electrică și calculatoare / 60 / Inginer

2. Date despre disciplină

2.1a Denumirea disciplinei/Categoria formativă[4]			Programare orientată pe obiecte / DS				
2.1b Denumirea disciplinei în limba engleză			Object Oriented Programming				
2.2 Titularul activităţilor de curs			Prof.dr.ing. Pănoiu Manuela				
2.3 Titularul activităţilor aplicative[5]			Șef lucr. dr. Ing. Ghiormez Loredana				
2.4 Anul de studii[6]	II	**2.5** Semestrul	4	**2.6** Tipul de evaluare	E	**2.7** Regimul disciplinei[7]	DOB

3. Timp total estimat - ore pe semestru: activități didactice directe (asistate integral sau asistate parțial) și activități de pregătire individuală (neasistate) [8]

3.1 Număr de ore asistate integral/săptămână	4 , format din:	**3.2** ore curs	2	**3.3** ore seminar/laborator/proiect	2
3.1* Număr total de ore asistate integral/sem.	56 , format din:	**3.2*** ore curs	28	**3.3*** ore seminar/laborator/proiect	28
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	3,14 , format din:	ore documentare suplimentară în bibliotecă, pe platformele electronice de specialitate și pe teren			1
		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			1.14
3.7* Număr total de ore activități neasistate/ semestru	44 , format din:	ore documentare suplimentară în bibliotecă, pe platformele electronice de specialitate și pe teren			14
		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			16
3.8 Total ore/săptămână [9]	7,14				
3.8* Total ore/semestru	100				
3.9 Număr de credite	4				

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

4.1 de curriculum	• Programarea calculatoarelor și limbaje de programare 2
4.2 de rezultatele învățării	•

5. **Condiţii** (acolo unde este cazul)

5.1 de desfăşurare a cursului	• Sală de curs echipată cu videoproiector și conexiune la Internet.
5.2 de desfăşurare a activităţilor practice	• Sală de laborator echipată cu cu videoproiector, tablă pentru scris și computere cu conexiune la internet.

6. **Rezultatele învățării** la formarea cărora contribuie disciplina

Cunoștințe	• C1. Studentul/absolventul identifică și descrie concepte, principii și metode de bază din matematică, fizică, economie și informatică. • C8. Studentul/absolventul identifică și descrie principiile utilizate în programare și modelare, simulare, baze de date, inteligență artificială, programare în timp real, învățare automată și rețele neuronale. .
Abilități	• A1.Studentul/absolventul ajustează proiectele de produse sau de părți de produse astfel încât acestea să îndeplinească cerințele. Studentul/absolventul creează și/sau execută un plan sau specificație pentru proiectarea unor sisteme industriale, materiale, produse sau un plan de producție, bazate pe concepte de design estetic și/sau funcțional. • A2. Studentul/absolventul specifică proprietăți tehnice ale bunurilor, materialelor, metodelor, proceselor, serviciilor, sistemelor, software-ului și funcționalităților, prin identificarea și răspunsul la nevoile particulare care urmează să fie satisfăcute în funcție de cerințele clienților. • A10. Studentul/absolventul selectează și aplică metodele actuale de modelare, calcul, proiectare și testare pentru specializarea lor.
Responsabilitate și autonomie	RA1. Studentul/absolventul aplică valorile eticii și deontologiei profesiei de inginer. RA2. Studentul/absolventul practică raționamentul logic, evaluarea și autoevaluare în luarea deciziilor. •

7. Obiectivele disciplinei (asociate rezultatelor învățării de la punctul 6)

- Obiectivul acestei discipline îl reprezintă însuşirea conceptelor specifice unui limbaj de programare orientat pe obiecte, cum ar fi: tip abstract de date, clasă, obiect, încapsulare, moştenire, derivare şi polimorfism.

Obiectivele specifice ale acestei discipline sunt:

- dezvoltarea deprinderilor de a rezolva probleme de dimensiuni mici şi medii într-o manieră orientată pe obiecte;
- dezvoltarea deprinderilor de a evidenţia diferenţa între proiectarea procedurală şi proiectarea orientată pe obiecte;
- dezvoltarea deprinderilor de a să înţelege rolul moştenirii, polimorfismului, legării dinamice şi a structurilor generice în dezvoltarea unor programe reutilizabile;
- dezvoltarea deprinderilor de a implementa programe în limbajul de programare C++ de dimensiuni mici/medii;
- dezvoltarea deprinderilor de a utiliza clase scrise de alţi programatori

8. Conţinuturi[10]

8.1 Curs	Număr de ore	Metode de predare[11]
1. Introducere în programarea orientată pe obiect 1.1. Declararea şi implementarea unei clase. Pointerul this 1.2. Protejarea informaţiilor din cadrul obiectelor. Încapsulare 1.3. Funcţii membre inline	2	Expunerea informațiilor esențiale pe videoproiector. Detalierea informațiilor expuse Demonstrarea utilizând un mediu de programare C/C++ Conversaţia
2. Crearea, intanţierea, iniţializarea şi distrugerea obiectelor 2.1. Liste de instanţieri 2.2. Constructori. Tipuri de constructori 2.3. Destructori 2.4. Manevrarea dinamică a obiectelor	2	
3. Elemente caracteristice claselor 3.1. Date şi funcţii membre statice 3.2. Obiecte constante. Funcţii membre constant 3.3. Funcţii şi clase prietene 3.4. Clase imbricate. Clase locale 3.5. Transmiterea obiectelor către funcţii. Returnarea obiectelor de către funcţii	4	
4. Moştenire şi polimorfism 4.1. Derivarea claselor. Reutilizarea funcţiilor din clase derivate 4.2. Polimorfismul. Supraîncărcarea şi supradefinirea funcţiilor.	6	

Funcţii virtuale 4.3. Protecţia informaţiei într-o ierarhie de clase 4.4. Clase virtuale. Clase abstracte 4.5. Supraîncărcarea operatorilor		
5. Tratarea excepţiilor. Sistemul de intrare/ieşire din C++ 5.1. Declararea excepţiilor. Ierarhii de excepţii 5.2. Ierarhia de clase corespunzătoare stream-urilor 5.3. Stream-uri predefinite în C++ 5.4. Redirectări. Supraîncărcarea operatorilor << şi >> 5.5. Operaţii de intrare/ieşire cu fişiere în C++	4	
6. Programare generică în C++. Biblioteca standard C++ 6.1. Funcţii şi clase generice în C++ 6.2. Bibiloteca standard C++ 6.2.1. Containere 6.2.2. Iteratori 6.2.3. Algoritmi 6.2.4. Functori. Clase de functori	6	
7. Biblioteca Qt 7.1. Arhitectura Qt. Biblioteca de clase Qt 7.2. QtCore 7.3. QtGui	4	

Bibliografie[12] 1. Manuela Pănoiu, Anca-Elena Iordan, Programare orientată pe obiecte în C++, Editura Politehnica, Timişoara, 2014
2. Liviu Negrescu, Limbajele C şi C++ pentru începători, Volumul II – Limbajul C++, Editura Albastră, Cluj-Napoca, 2006
3. Scott Meyers, STL – Biblioteca programatorului, Editura Teora, Bucureşti, 2002
4. Constantin Gălățan, C++ - Introducere în Standard Template Library, Editura All, București, 2008

8.2 Activităţi aplicative[13]	Număr de ore	Metode de predare
1. Analiza, proiectarea şi implementarea de aplicaţii care utilizează tipuri de date abstracte definite prin intermediul structurilor. 2. Analiza, proiectarea şi implementarea de aplicaţii care utilizează tipuri de date abstracte definite prin intermediul claselor	4	Verificare cunoștințe din tematica laboratorului Se va utiliza exercițiul la tablă și implementarea programului utilizând calculatorul.
3. Analiza, proiectarea şi implementarea de aplicaţii care utilizează constructori și destructori	2	
4. Analiza, proiectarea şi implementarea de aplicaţii care utilizează date și funcții statice. Funcții prietene	2	
5. Analiza, proiectarea şi implementarea de aplicaţii care utilizează compunerea de obiecte. 6.Analiza, proiectarea şi implementarea de aplicaţii care utilizează conceptul de moștenire.	4	
7. Analiza, proiectarea şi implementarea de aplicaţii care utilizează funcţii virtuale pure. Clase abstracte	2	Verificare cunoștințe din tematica laboratorului Se va utiliza exercițiul la tablă și implementarea programului utilizând calculatorul.
8. Analiza, proiectarea şi implementarea de aplicaţii care utilizează supraîncărcarea operatorilor	4	Verificare cunoștințe din tematica laboratorului Se va utiliza exercițiul la tablă și implementarea programului utilizând calculatorul.
9. Analiza, proiectarea şi implementarea de aplicaţii care utilizează ierarhii de clase pentru operaţii de intrare/ieşire	2	Verificare cunoștințe din tematica laboratorului Se va utiliza exercițiul

		la tablă și implementa-rea programului utilizând calculatorul.
10. Analiza, proiectarea şi implementarea de aplicaţii prin reutilizarea codului sursă cu ajutorul claselor șablon	6	Verificare cunoștințe din tematica laboratorului Se va utiliza exercițiul la tablă și implementa-rea programului utilizând calculatorul.
11. Analiza, proiectarea şi implementarea de aplicaţii prin reutilizarea codului sursă cu ajutorul containerelor secvențiale STL	2	Verificare cunoștințe din tematica laboratorului Se va utiliza exercițiul la tablă și implementa-rea programului utilizând calculatorul.
Bibliografie[14] 1. Ghiormez L., Laborator în format electronic – Programare orientată pe obiecte, Campusul virtual al UPT, https://cv.upt.ro/course/view.php?id=2617 sau intranet facultate \\fs\cursuri\PROGRAMAREA CALC SI LIMBAJE DE PROGRAMARE - II IIND + II IEC 2025 2 Ghiormez L., Pănoiu M., Programarea calculatoarelor în limbajul C/C++, Ed. POLITEHNICA, Timișoara, 326 pg., 2022, ISBN: 978-606-35-0376-4 3. Iordan A., Cristea D., Berdie A., Programarea calculatoarelor, îndrumător de laborator, Editura Mirton, Timişoara, 2004 4. L. Negrescu, Limbajele C şi C++ pentru începători, Volumul I – Limbajul C, Editura Albastră, Cluj-Napoca, 2009		

9. Evaluare

Tip activitate	**9.1** Criterii de evaluare[15]	**9.2** Metode de evaluare	**9.3** Pondere din nota finală
9.4 Curs	- Înțelegerea principiilor programării orientate pe obiecte în Java. - Cunoașterea sintaxei și a mecanismelor limbajului Java. - Capacitatea de proiectare logică a aplicațiilor orientate pe obiecte .	Examen final (scris sau pe calculator) pentru verificarea cunoștințelor teoretice și aplicative	66%
9.5 Activităţi aplicative	**S:**		
	L: Abilități în analiza, proiectarea și implementarea aplicațiilor de laborator. Teme de casă și/sau în timpul desfășurării activității practice	Oral – aplicații utilizând calculatorul	34 %
	P[16]**:**		
	Pr:		
9.6 Standard minim de performanţă (se prezintă cunoștințele minim necesare pentru promovarea disciplinei și modul în care se verifică stăpânirea lor[17])			
• ***Laborator:*** Studentul trebuie să dețină capacitatea de a rescrie în forma corectă o declarație de clasă C++ ce modelează atributele fizice/electrice ale unui dispozitiv pentru a preveni stările invalide ale sistemului, să ruleze și să depaneze clasa implementată în mediul de dezvoltare (IDE) și să utilizeze documentația furnizată pentru rezolvarea cerinței (C3, A3, A5, RA5). Studentul trebuie să dețină capacitatea de testare și tratare a cazurilor de eroare (ex.: împărțire la zero, depășire de memorie, valori fizice imposibile) conștientizând că un software destinat ingineriei electrice defectuos poate produce daune materiale sau umane (RA7). • ***Curs***: Realizarea unei aplicații C++ funcționale, care implementează clase și funcții proprii și demonstrează înțelegerea principiilor POO (încapsulare, moștenire, polimorfism)) • *Pentru a promova disciplina trebuie ca ambele activități (curs și laborator) să fie promovate cu minim 5.*			

Data completării	**Titular de curs (semnătura)**	**Titular activităţi aplicative (semnătura)**
10.09.2026		

Director de departament (semnătura)	Data avizării în Consiliul Facultăţii[18]	Decan (semnătura)
	23.09.2026	