Ficha de Unidade Curricular – (Versão A3ES 2018-2023)

1. Caracterização da Unidade Curricular.

1.1. Designação da unidade curricular ***(1.000 carateres).***
Programação III

1.2. Sigla da área científica em que se insere ***(100 carateres).***
IC

1.3. Duração[1] ***(100 carateres).***
Semestral

1.4. Horas de trabalho[2] ***(100 carateres).***
162

1.5. Horas de contacto[3] ***(100 carateres).***
T: 22,5h; TP: 22,5H; PL: 22,5h

1.6. ECTS ***(100 carateres).***
6

1.7. Observações[4] ***(1.000 carateres).***
Obrigatória (UC comum com outros cursos)

1.7. Remarks ***(1.000 carateres).***
Common with other courses

2. Docente responsável e respetiva carga letiva na Unidade Curricular ***(preencher o nome completo) (1.000 carateres).***
Maria Manuela da Silva Veiga Torres de Sousa – 135 horas

3. Outros docentes e respetivas cargas letivas na unidade curricular ***(1.000 carateres).***

4. Objetivos de aprendizagem (conhecimentos, aptidões e competências a desenvolver pelos estudantes). ***(1.000 carateres).***

Os estudantes que terminam com sucesso esta unidade curricular serão capazes de:

1. Definir e usar: classes derivadas, classes que representem Estruturas de Dados dinâmicas, e genéricos.
2. Definir e usar expressões lambda para passar funcionalidade como argumento a outro método.
3. Usar interfaces gráficas nas aplicações construídas.
4. Construir aplicações simples usando o paradigma da Programação Orientada por Objetos.
5. Testar e corrigir as aplicações construídas, usando as ferramentas de desenvolvimento adequadas.
6. 5.Escrever relatórios onde se justifica a hierarquia de classes e a estrutura de dados usada nas aplicações construídas.

4. Intended learning outcomes (knowledge, skills and competences to be developed by the students). ***(1.000 characters).***

Students who successfully complete this course unit should be able to:

1. Define and use derived classes, and classes that represent dynamic data structures.
2. Define and use lambda expressions to pass functionality as an argument to another method.
3. Use Graphical User Interface in applications build.
4. Build small applications using the paradigm of Object Oriented Programming.
5. Test and repair applications using the development tools appropriate.
6. Write reports to justifying the class hierarchy and the data structure used in the applications built.

5. Conteúdos programáticos ***(1.000 carateres).***

I. Herança e polimorfismo: classes derivadas; classes abstratas; interfaces; ligação dinâmica.
II. Ficheiros binários e de texto. Tratamento de exceções.
III. Estruturas de dados dinâmicas: vetores, listas ligadas, conjuntos e contentores associativos. Iteradores.

Comparadores. Genéricos.

IV. Expressões lambda.

V. Introdução à interface gráfica: *event-driven*; *listeners*; *layout managers*; *Model-View-Controller*.

5. Syllabus ***(1.000 characters).***

I. Inheritance and polymorphism: derived classes; abstract classes; interfaces; dynamic binding.

II. Text and binary files. Exception handling.

III. Dynamic Data Structures: vectors, linked lists, sets and maps. Iterators. Comparators. Generics.

IV. Lambda expression.

V. Introduction to GUI: event-driven programming; listeners; layout managers; Model-View-Controller.

6. Demonstração da coerência dos conteúdos programáticos com os objetivos de aprendizagem da unidade curricular ***(1.000 carateres).***

Esta unidade curricular introduz os conceitos e o vocabulário fundamental dos paradigmas da programação orientada por objetos (I) e da programação *event-driven* (V), concretizados na linguagem Java (1), (3) e (4). A hierarquia de classes dos *streams* (II) e a hierarquia de classes das estruturas de dados dinâmicos da *framework* de coleções do Java (III) são usadas para consolidar os conceitos transmitidos (1).

Esta unidade curricular introduz também o estudo das estruturas de dados dinâmicas (III). A análise do desempenho das representações, em *array*, lista ligada, árvores e tabelas de *hash* permite selecionar a estrutura de dadas adequada (1), (4) e (6).

Nesta unidade curricular usa-se expressões lambda (IV) quando os problemas requerem passar funcionalidade como parâmetro a outro método (2).

6. Evidence of the syllabus coherence with the curricular unit's intended learning outcomes ***(1.000 characters).***

This curricular unit introduces the fundamental concepts and vocabulary of Object Oriented Programming (I), and event-driven programming (V), implemented in Java (1), (3) and (4). The class hierarchy of streams (II) and the class hierarchy of dynamic data structures in the Java collections framework (III) are used to consolidate the concepts transmitted (1).

This curricular unit also introduces the study of dynamic data structures (III). Analysis of the performance of representations, in array, linked list, trees and hash tables, allows selecting the appropriate data structure (1), (4) and (6).

In this curricular unit are used lambda expressions (IV) when the problems require pass functionality as an argument to another method (2).

7. Metodologias de ensino (avaliação incluída) ***(1.000 carateres).***

Ensino teórico-prático, estando previstas 30 aulas durante o semestre a que correspondem 67,5 horas de contacto (15 aulas de 3 horas e 15 de 1,5 horas). O tempo total de trabalho do estudante é de 162 horas. As aulas interativas destinam-se à apresentação dos temas e de exemplos práticos de aplicação (aprendizagem baseada em casos). Os tópicos principais são ainda explorados através da realização de trabalhos práticos para desenvolver pequenas aplicações em Java (aprendizagem baseada na resolução de problemas).

Os resultados de aprendizagem (1), (2) e (3) são avaliados individualmente através de dois testes escritos realizados durante o semestre ou um exame final. Durante o acompanhamento dos trabalhos de grupo são avaliados os resultados de aprendizagem (4) e (5). Os resultados de aprendizagem (4) e (6) são avaliados na discussão final dos trabalhos de grupo, onde é discutida a qualidade das soluções.

A nota final é a média aritmética da nota escrita (E- média de testes ou exame final) e da nota da discussão (D - discussão dos trabalhos realizados ao longo do semestre, dando especial relevância ao último)).

7. Teaching methodologies (including assessment) ***(1.000 characters).***

Theoretical and practical teaching is planned during the semester in 30 lectures that correspond to 67.5 of contact hours (15 lectures of 3 hours and 15 of 1.5 hours). The total student working hours is 162. Interactive lectures are used for presentation of topics and practical examples (case-based learning). The main topics are further explored through practical work to develop small applications in Java (problem-based learning).

Learning outcomes (1), (2) and (3), are individually assessed through two written tests during the semester or a final written exam. The learning outcomes (4) and (5) are assessed during the monitoring of group work. Learning outcomes (4) and (6) are evaluated in the final discussion of work group, where we discuss the quality of the solutions.

The grade is the arithmetic mean of the written note (W-test average or final exam) and note of the discussion

(D - discussion of the work carried out during the semester, giving special emphasis to the last one).

8. Demonstração da coerência das metodologias de ensino com os objetivos de aprendizagem da unidade curricular ***(3.000 carateres).***

O conhecimento dos conceitos fundamentais da programação orientada por objetos e da programação *event-driven* é obtido através de aulas interativas e respetivos elementos de apoio, e da realização de uma aplicação, com a participação dos alunos, que necessita de especializar comportamentos (polimorfismo), reutilizar código (herança), armazenar dados (estruturas de dados dinâmicas), e usa uma interface com o utilizador textual ou gráfica (1-4).

A competência para desenvolver boas práticas de desenho e codificação de forma a produzir aplicações usando o paradigma da programação orientada por objetos e interfaces gráficas (3-6) é desenvolvida através da realização de trabalhos com supervisão e da sua avaliação critica.

8. Evidence of the teaching methodologies coherence with the curricular unit's intended learning outcomes ***(3.000 characters).***

Knowledge of the fundamental concepts of Object Oriented Programming and event-driven programming is achieved through interactive lessons, support elements, and the building of an application, with the participation of students, that need to specialize behaviors (polymorphism), code reuse (inheritance), implement lambda expressions (, data store, and uses a user interface textual or graphics (1-4).

The ability to develop good practice in design and coding to produce applications using the paradigm of object oriented programming and graphics user interface (3-6), is developed by performing work under supervision and their critical evaluation.

9. Bibliografia de consulta/existência obrigatória ***(1.000 carateres).***

W. Savitch , *Java: An Introduction to Problem Solving and Programming, 8th Edition* , 2018 | Pearson

---

[1] Anual, semestral, trimestral, …

[2] Número total de horas de trabalho.

[3] Discriminadas por tipo de metodologia adotado (T - Ensino teórico; TP - Ensino teórico-prático; PL - Ensino prático e laboratorial; TC - Trabalho de campo; S - Seminário; E - Estágio; OT - Orientação tutorial; O - Outro).

[4] Assinalar sempre que a unidade curricular seja optativa.