Ficha de Unidade Curricular A3ES
Programação III
Licenciatura em Engenharia Eletrónica e Telecomunicações e de Computadores

2024-25

1. Caracterização da Unidade Curricular

1.1 Designação
[3661] Programação III / Programming III

1.2 Sigla da área científica em que se insere
IC

1.3 Duração
Unidade Curricular Semestral

1.4 Horas de trabalho
160h 00m

1.5 Horas de contacto
Total: 67h 30m das quais T: 45h 00m | P: 22h 30m

1.6 ECTS
6

1.7 Observações
Unidade Curricular Obrigatória
Unidade Curricular comum ao(s) curso(s) de LEIRT

2. Docente responsável [715] Maria Manuela da Silva Veiga Torres de Sousa

3. Docentes e respetivas cargas letivas na unidade curricular [715] Maria Manuela da Silva Veiga Torres de Sousa | Horas Previstas: 135 horas

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

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

1. Definir e usar: classes derivadas, classes que representem estruturas de dados dinâmicos, e genéricos.

2. Definir e usar expressões lambda para passar funcionalidade como argumento a outro método.

3. Construir aplicações simples usando o paradigma da Programação Orientada por Objetos e *frameworks* para construção de interfaces gráficas.

4. Testar e corrigir as aplicações construídas, usando as ferramentas de desenvolvimento adequadas.

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)

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. Build small applications using the paradigm of Object Oriented Programming and frameworks for building graphics interfaces.

4. Test and repair applications using the development tools appropriate.

5. Write reports to justifying the class hierarchy and the data structure used in the applications built.

5. Conteúdos programáticos

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. Interfaces funcionais e expressões lambda.

V. Introdução à construção de aplicações com interface gráfica : programação *event-driven; listeners* ; *layout managers* ; padrão *Model-View-Controller* .

5. Syllabus

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. Functional interfaces and lambda expression.

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

6. Demonstração da coerência dos conteúdos programáticos com os objetivos de aprendizagem da unidade curricular

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) e (3). A hierarquia de classes dos *streams* (II) e a hierarquia de classes da *framework* de coleções do Java (III) são usadas para consolidar os conceitos transmitidos (1).

O estudo das estruturas de dados dinâmicas (III) e a análise do desempenho das representações, em *array* , lista ligada, árvores e tabelas de *hash* permite selecionar a estrutura de dadas adequada (1), (3) e (5).

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

This curricular unit introduces the fundamental concepts and vocabulary of Object Oriented Programming (I), and event-driven programming (V), implemented in Java (1) and (3). 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).

The study of dynamic data structures (III) and the analysis of the performance of representations, in array, linked list, trees and hash tables, allows selecting the appropriate data structure (1), (3) and (5).

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)

Ensino teórico e 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). As aulas interativas destinam-se à apresentação dos temas e de exemplos de aplicação. Os tópicos principais são ainda explorados através da realização de trabalhos práticos, pedagogicamente fundamentais, para desenvolver pequenas aplicações em Java (aprendizagem baseada na resolução de problemas).

Os resultados de aprendizagem (1) e (2) são avaliados individualmente através de dois testes escritos (1º realizado durante o semestre, o 2º na 1ª data de exame) ou um exame final. Durante o acompanhamento dos trabalhos de grupo são avaliados os resultados de aprendizagem (3) e (4). Os resultados de aprendizagem (3) e (5) 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 da prova escrita e da nota da discussão dos trabalhos.

7. Teaching methodologies (including assessment)

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). Interactive lectures are used for presentation of topics and practical examples. The main topics are further explored through practical work, pedagogically fundamental, to develop small applications in Java (problem-based learning).

Learning outcomes (1) and (2), are individually assessed through two written tests (1st during the semester, 2nd on regular exam) or a final written exam. The learning outcomes (3) and (4) are assessed during the monitoring of group work.Learning outcomes (3) and (5) 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 (test average or final exam) and note of the individual discussion.

8. Demonstração da coerência das metodologias de ensino com os objetivos de aprendizagem da unidade curricular

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

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	W. Savitch , *Java: An Introduction to Problem Solving and Programming, 8th Edition* , 2021 | Pearson
10. Data de aprovação em CTC	2024-07-17 2024-07-17
11. Data de aprovação em CP	2024-06-26 2024-06-26