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

1. Caracterização da Unidade Curricular

1.1. Designação da unidade curricular

Modelação e Programação (MoP)

1.2. Sigla da área científica em que se insere

Engenharia Informática (INF)

1.3. Duração

Semestral

1.4. Horas de trabalho

Total de Horas: 162

1.5. Horas de contacto

Total de 67,5 horas = (T → 15) + (TP → 7,5) + (PL → 45)

1.6. ECTS

6

1.7. Observações

O ensino da disciplina é essencialmente de cariz prático, com o intuito de maximizar o número de horas que o aluno passa a programar. Os conceitos aprendidos nas aulas teóricas têm aplicação nas aulas práticas que se seguem, onde o aluno tem que resolver problemas, criando pequenas aplicações, propositadamente desenhadas para testar e consolidar esses conceitos.

1.7. Remarks

This is mainly a hand-on course, with the goal of maximizing the number of hours each student spends programming. The concepts learned in theory classes have direct application in the following practice sessions where the student must solve problems, by creating small applications, specifically tailored for testing and consolidating these concepts.

2. Docente responsável e respetiva carga letiva na Unidade Curricular

Pedro Viçoso Fazenda, com a carga lectiva de 9horas/semana

3. Outros docentes e respetivas cargas letivas na unidade curricular

António Gelásio Frazão Isidro Teófilo, com a carga lectiva de 4,5 horas /semana
Carlos Jorge de Sousa Gonçalves, com a carga lectiva de 4,5 horas /semana

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) Usar a Unified Modeling Language (UML) como ferramenta de modelação, nomeadamente o seu diagrama de classes.
(2) Aplicar a modelação e programação orientada a objetos para construir aplicações.
(3) Criticar a estrutura de aplicações modeladas com objetos.
(4) Aplicar a modelação e programação orientada a eventos para construir aplicações.
(5) Desenvolver aplicações gráficas, incluindo aplicações que utilizem recursos multimedia.
(6) Usar ferramentas para edição, execução e depuração de aplicações.

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

When students complete this course, the will be able to:
(1) Use the Unified Modelling Language (UML) as a tool for modelling class diagrams.
(2) Creating object-oriented applications.
(3) Model applications using the object-oriented programming paradigm.
(4) Create event-driven programming applications.
(5) Create graphical user interface applications, including applications that use multimedia resources.
(6) Use and integrated development environment to edit, execute and debug applications.

5. Conteúdos programáticos

I. Introdução ao Java.
II. Noções de classe, instância, atributo e método.
III. Noções de herança e conceitos associados: derivação, polimorfismo, ligação dinâmica, classes abstratas e interfaces.
IV. Modelação de cenários utilizando o diagrama de classes do UML.
V. Exceções, sua definição, lançamento e tratamento.
VI. Entrada e saída de dados.
VII. Estruturas de dados dinâmicas (com especial enfoque nas listas e iteradores) e classes genéricas.
VIII. Conceitos de programação orientada a eventos: evento, recetor de evento e envio e receção de eventos.
IX. Desenvolvimento de aplicações baseadas em eventos.
X. Apresentação da interface gráfica Swing e seus principais componentes. O padrão de desenho Model View Controller (MVC).
XI. Conceitos de programação com recursos multimédia.
XII. Desenvolvimento de aplicações Swing com recursos multimédia.

5. Syllabus

I. Introduction to the Java programming language.
II. Notion of class, instance, method and attribute.
III. Notion of inheritance and behavioural subtyping, polymorphism, dynamic binding, abstract classes and interfaces.
IV. Modelling applications using UML class diagrams.
V. Exceptions and error handling.
VI. Data input and output.
VII. Dynamic Data Structures (Lists, Arrays, Sets, Queues, Hash-based structures, etc.) and generic classes.
VIII. Concept of event-driven programming: event, handler/listener and event dispatching.
IX. Development of event-driven applications.
X. Java Swing widget toolkit graphical user interface. The Model View Controller (MVC) design pattern.
XI. Creating interactive programs with multimedia resources.
XII. Developent of Swing applications with multimedia resources.

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

Esta unidade curricular lida com os conceitos e práticas da modelação e conceção de aplicações, nomeadamente o paradigma da programação orientada por objetos, o paradigma da programação orientada por eventos e a programação com recursos multimédia. Estes conceitos são concretizados na linguagem Java nomeadamente na sua componente estrutural, explorando o paradigma dos objetos, e na sua componente dinâmica, explorando os modelos de execução sequencial e de execução por eventos.

6. Evidence of the syllabus coherence with the curricular unit's intended learning outcomes

This curse is mainly focused on practical concepts concerning applications modelling and programming, particularly focused on the object-oriented programming paradigm, event-driven applications and programming with graphical user interfaces and multimedia resources. These concepts are tested and carried out using the Java programming language for learning structure, taking advantage of its object-oriented paradigm, and its dynamic behaviour, exploring the sequential and event-driven execution models.

7. Metodologias de ensino (avaliação incluída)

A metodologia de ensino desenvolve-se em três componentes:

T – 15 horas de exposição teórica.
TP – 7,5 horas. Por cada tema teórico são apresentados exemplos e resolvidos exercícios.
PL – 45 horas de contato de prática laboratorial. Os conceitos teóricos são consolidados em aulas laboratoriais através da implementação de exercícios realizados em grupo.

Os objetivos de aprendizagem (1) a (6) são avaliados através de exame escrito e de trabalhos práticos, com datas de entrega dos relatórios bem determinadas, executados e avaliados em grupo (com nota individual) numa discussão final. A componente teórica contribui 40% para a nota final (com nota mínima de aprovação de 9.5) e a componente prática contribui com 60%.

7. Teaching methodologies (including assessment)

The lecture method of teaching divides into the following hours:

T – 15 hours of theoretical exposition.
TP – 7,5 hours. For each theme, theoretical examples are presented, and some exercises are solved.
PL - 45 horas of programming hours in the lab. The theoretical concepts are consolidated with several projects contain several exercises each, where students are incentivized to work as team (maximum two students/per group).

The learning objectives stated in (1) to (6) are evaluated with a final written exam and with the outcome of each of the practical projects (and associated reports) that are delivered throughout the semester on different dates. Projects are evaluated as team-work, with final project discussions and individual classifications. The written exam contributes for 40% of the final grade (requiring a minimum grade of 9.5) and the evaluation of the practical projects is worth 60%.

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

Nas aulas é dado o programa correspondente aos objetivos de aprendizagem (1) a (6). São apresentados exemplos e resolvidos exercícios. Nas aulas laboratoriais pretende-se que os estudantes antecipem soluções,

para isso, é fornecido antecipadamente um guia laboratorial para cada um dos trabalhos. Tendo em conta o cumprimento dos objectivos da disciplina, os trabalhos laboratoriais focam-se essenciamente em: a) Introdução à programação Java e estruturas de dados; b) os tópicos abordados em (3), divididos em objectos e instâncias e herança e polimorfismo; c) os tópicos abordados em (5) e (6), com uma aplicação final (normalmente um jogo) com várias alíneas obrigatórias e outras opcionais. Tendo em consideração o cumprimento dos objetivos (3), (5) e (6), as soluções propostas pelos estudantes são discutidas no âmbito da turma. Na discussão final é avaliado o trabalho, realizado autonomamente em grupo, com particular destaque para os relatórios, aproveitando a oportunidade para salientar aspetos manifestados nos objetivos de aprendizagem (3), (5), e (6) que sejam considerados oportunamente relevantes.

8. Evidence of the teaching methodologies coherence with the curricular unit's intended learning outcomes

In class the syllabus of the course is taught to fulfil the learning objectives stated in (1) to (5). Examples are presented in class with hands-on exercises. In the lab, students are expected to anticipate solutions by being receiving, in anticipation, a lab guide for each practical project that they must fully complete at home and in the following TP classes.
Attending to the objectives of the course, these practical projects are essentially divided according to the following items:

a) Introduction to Java programming and data structures;
b) The topics referred in (3), divided in object-oriented programming and inheritance, polymorphism, abstract classes and interfaces;
c) The topics referred in (5) and (6), with the development of a fully functional final application (usually a game) with several mandatory items and a few optional items.

Considering the execution of the objectives stated in (3), (5) and (6), all solutions proposed by the students are discussed in class. In the final discussion of each project each individual team-project is evaluated (with a detailed review of the documentation and reporting). The outcome of this evaluation is highly dependent on what each student learned in (3), (5) and (6).

9. Bibliografia de consulta/existência obrigatória

Savitch W, Java: An Introduction to Problem Solving and Programming (8th edition), Pearson, 2017.