Ficha de Unidade Curricular A3ES
Linguagens e Ambientes de Execução
Licenciatura em Engenharia Informática e de Computadores
2024-25

---

1. Caracterização da Unidade Curricular

1.1 Designação
[4173] Linguagens e Ambientes de Execução / Languages and Managed Runtimes

1.2 Sigla da área científica em que se insere
IC

1.3 Duração
Unidade Curricular Semestral

1.4 Horas de trabalho
162h 00m

1.5 Horas de contacto
Total: 67h 30m das quais TP: 67h 30m

1.6 ECTS
6

1.7 Observações
Unidade Curricular Obrigatória

---

2. Docente responsável [1619] Fernando Miguel Santos Gamboa Lopes de Carvalho

---

3. Docentes e respetivas cargas letivas na unidade curricular Não existem docentes definidos para esta unidade curricular

---

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. Comparar e utilizar diferentes construções comuns em linguagens de programação modernas, enquadrando diferentes paradigmas de programação, e o seu suporte no ambiente de execução.

2. Entender os principais constituintes de um ambiente de execução para linguagens de alto nível, e saber comparar diferentes abordagens de sistemas de tipos destes ambientes.

3. Usar metadados em tempo de execução (reflexão) para examinar tipos e usar metaprogramação para analisar e transformar programas em tempo de execução.

4. Analisar o desempenho de programas *managed* e usar eficientemente o suporte automático de gestão de memória (*garbage collection*).

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

A student completing this course unit should be able to:

1. Compare and use different common constructs in modern programming languages, mapping different programming paradigms, and their support in the execution environment.

2. Understand the main constituents of managed runtimes, and know how to compare different type systems approaches in these environments.

3. Use runtime metadata (reflection) to examine types and use metaprogramming to analyze and transform programs at runtime.

4. Analyze the performance of managed programs and efficiently use automatic memory management support (garbage collection).

5. Conteúdos programáticos

I. Principais construções de linguagens suportadas em ambientes de execução para linguagens de alto nível (*managed runtimes*) e a sua contextualização em diferentes paradigmas de programação, tendo como principal caso de estudo a máquina virtual Java.

II. Principais constituintes dos ambientes de execução *managed* , nomeadamente: *class loader* , *verifier* , *just-in-time compiler* , metadados e gestão de memória.

III. Comparação de sistemas de tipos para ambientes de execução *managed* quanto às regras de equivalência, compatibilidade e inferência.

IV. API de reflexão em Java e casos práticos de metaprogramação no desenvolvimento de software.

V. Modelo de pilha e registos. Análise das principais instruções bytecode Java apoiada em ferramentas de suporte à reflexão estrutural (e.g. ASM, Javassist).

VI. Introdução à análise de desempenho de programas Java e uso de ferramentas de monitorização da JVM (e.g. *jconsole*).

VII. Introdução aos algoritmos de *garbage collection na JVM* .

5. Syllabus

I. Main constructions of languages supported in managed runtimes, and their contextualization in different programming paradigms, using as main case study the Java Virtual Machine (JVM).

II. Main constituents of managed runtimes, namely: class loader, verifier, just-in-time compiler, metadata and memory management.

III. Comparison of type systems for managed runtimes regarding: equivalence rules, compatibility and inference.

IV. Java reflection API and metaprogramming case studies in software development.

V. Execution stack model and records. Analysis of the main Java bytecode instructions supported by tools for structural reflection (e.g. ASM, Javassist).

VI. Introduction to the performance analysis of programs written in Java, and the use of monitoring tools for the JVM (e.g. jconsole).

VII. Introduction to garbage collection algorithms in the JVM.

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

Esta unidade curricular identifica os principais problemas resolvidos por um ambiente de execução para linguagens de alto nível e o suporte que fornece ao desenvolvimento de aplicações. Em particular são analisadas diferentes construções disponíveis nestas linguagens e nos sistemas de tipos, usando como principal caso de estudo a linguagem Java.

A coerência entre os conteúdos programáticos e os objetivos da unidade curricular é a seguinte:

- O objetivo 1 é alcançado através dos conteúdos I e III;
- O objetivo 2 é alcançado através dos conteúdos II, III, e IV;
- Os conteúdos IV e V contribuem para o objetivo 3;
- Os conteúdos V, VI e VII pretendem concretizar o objetivo 4.

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

This course unit identifies the main problems solved by an execution environment for high-level languages and the support it provides for application development. In particular, it analyzes different constructs available in these languages and their type systems, using Java as the main case study.

The coherence between the program contents and the objectives of the course unit is as follows:

- Objective 1 is achieved through contents I and III;
- Objective 2 is achieved through contents II, III, and IV;
- Contents IV and V contribute to objective 3;
- Contents V, VI, and VII aim to achieve objective 4.

7. Metodologias de ensino (avaliação incluída)

O ensino teórico-prático é planeado durante o semestre em 30 aulas que correspondem a 67,5 horas de contato (15 aulas de 3 horas e 15 aulas de horas e meia). O total de horas de trabalho do aluno é de 162. As aulas destinam-se à apresentação e explicação dos tópicos e sua demonstração prática.

A unidade curricular tem avaliação distribuída com exame final. Os resultados da aprendizagem são avaliados individualmente através da prova escrita realizada no final do semestre (**PE**) e da avaliação do trabalho prático realizado durante o semestre (**TP**). Durante o acompanhamento do trabalho em grupo realizado em aulas práticas, os resultados da aprendizagem são avaliados juntamente com uma discussão final do trabalho prático (TP). A nota final é a média ponderada da prova escrita (PE) (50%) e do trabalho prático com discussão final (TP) (50%). Cada uma das componentes da avaliação (PE e CP) tem que ter uma avaliação superior ou igual a 9,50 valores.

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 lessons of 3 hours and 15 1.5 hours). The total student working hours is 162. The lectures are intended for presentation and explanation of the topics and their practical application demonstration.

The curricular unit has distributed assessment with a final exam. Learning outcomes are evaluated individually through the written test conducted at the end of the semester (PE), and through the evaluation of the practical work delivered during the semester (TP). During follow-up of group work performed in practical lectures, the learning outcomes are assessed and evaluated in a final work discussion (TP). The final grade is the weighted average of the written test (PE) (50%) and practical work and discussion (TP) (50%). Both PE and TP must have a final grade greater than or equal to 9,50 values.

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

O conhecimento dos aspectos teóricos da unidade curricular é obtido por meio de aulas teórico-práticas e de exercícios interativos. As aulas são complementadas com sessões práticas, onde os alunos devem resolver problemas usando as ferramentas de desenvolvimento que serão usadas na resolução de um projeto de desenvolvimento de software.

Este projeto inclui um conjunto de desafios que devem ser resolvidos através da aplicação do conhecimento adquirido.

A avaliação do projeto é baseada em uma discussão final que aborda as soluções desenvolvidas pelo aluno.

8. Evidence of the teaching methodologies coherence with the curricular unit´s intended learning outcomes	Knowledge of the theoretical aspects of the course unit is obtained through theoretical-practical classrooms and interactive exercises. Classrooms are complemented with practical sessions, where students must solve problems using the development tools that will be used in the resolution of a software development project. This project includes a set of challenges that must be solved by applying the acquired knowledge. The project evaluation is based on a final discussion that addresses the solutions developed by the student.
9. Bibliografia de consulta/existência obrigatória	" *The Managed Runtime Environment: Diving into the JVM with Kotlin* ", First Edition, Miguel Gamboa, http://leanpub.com/kotlinonjvm, 2024 " *The Well-Grounded Java Developer* ", Second Edition, Benjamin Evans, Jason Clark, and Martijn Verburg, October 2022, ISBN 9781617298875
10. Data de aprovação em CTC	2024-07-17
11. Data de aprovação em CP	2024-06-26