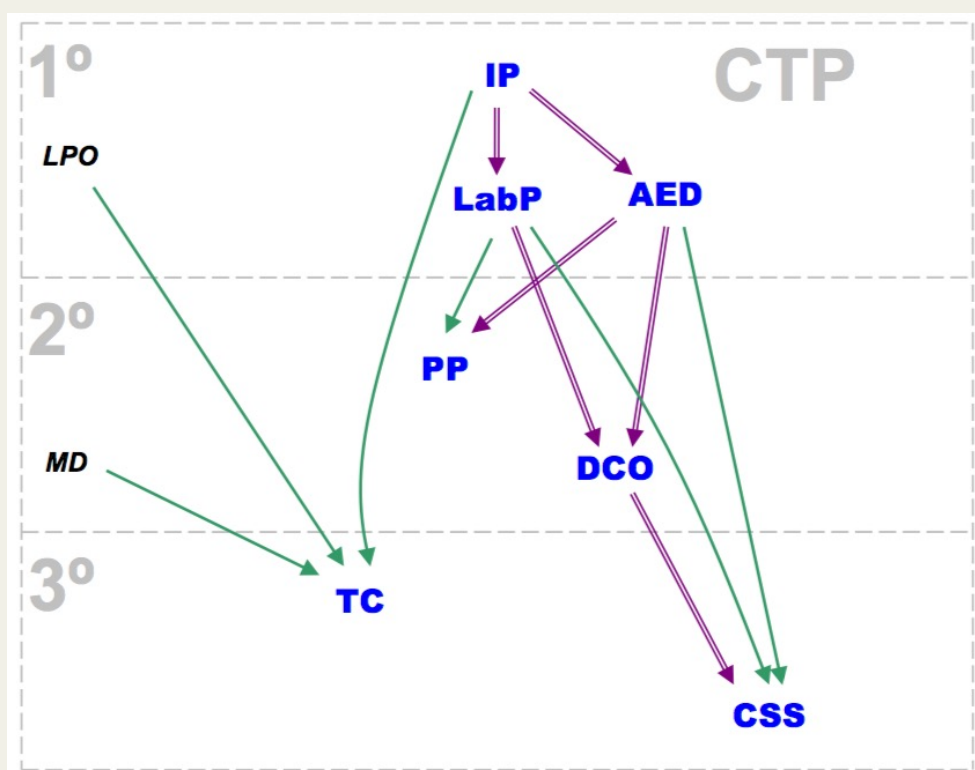


DESENHO CENTRADO EM OBJECTOS

Antónia Lopes

DCO @ LEI





Ciências
ULisboa | Informática

Programming-in-the-large

- **Escala do código**

MLOC

- **Desenvolvimento em equipa**

Muitas decisões exigem comunicação entre diferentes elementos

- **Período de desenvolvimento** — Meses ou anos

Planos têm que considerar *milestones*

- **Período de operação e evolução** — Anos

Os programas duram mais do que os programadores

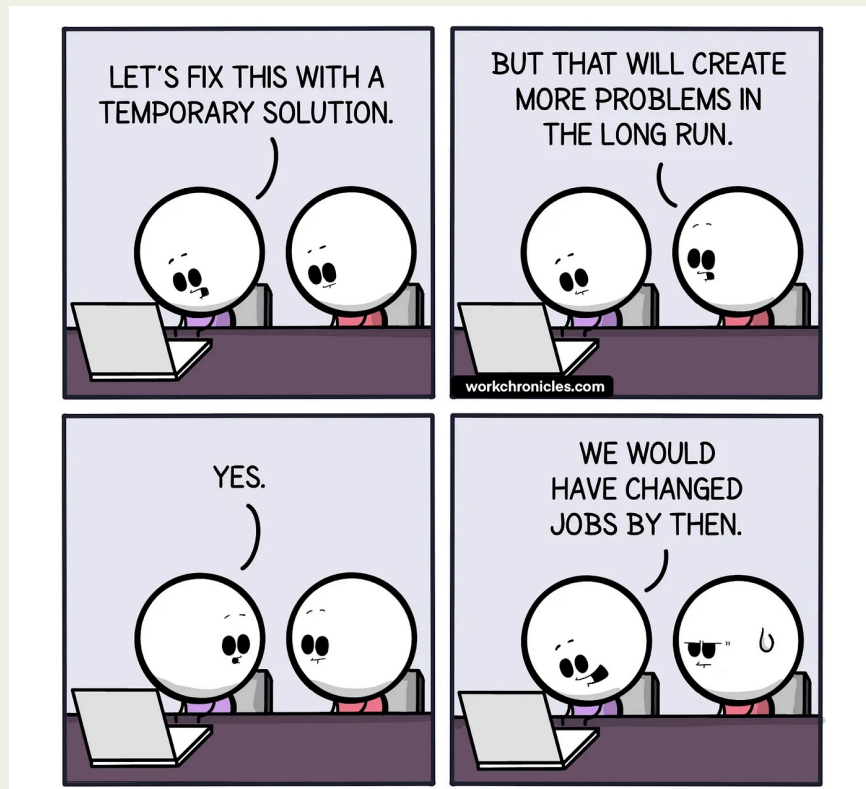
multi-person development of multi-version software

Orientação a Objetos

- Paradigma parcialmente motivado pela taxa de insucesso de utilização do paradigma imperativo no contexto do *programming-in-the-large*
- Problemas no **desenvolvimento**
 - derivados da **complexidade**
- Problemas na **manutenção**
 - derivados da dificuldade de lidar com o impacto das **mudanças**

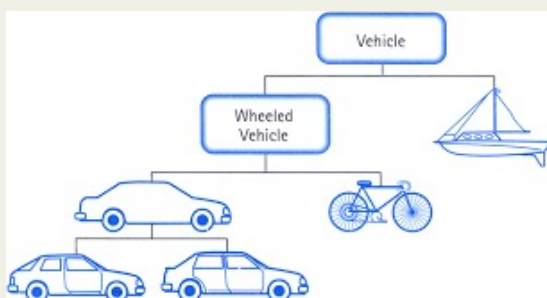
Mudança no Software

- A **mudança é inevitável** para os sistemas de software se manterem úteis
 - mudam os requisitos
 - muda o contexto de negócio
 - são revelados problemas que têm de ser reparados
 - ...
- A **gestão da mudança é difícil**
 - a complexidade dos programas
 - a ligação entre as várias partes
 - a dificuldade em manter o controlo sobre o todo
 - a impossibilidade de começar tudo de raiz
 - ...



Orientação a Objetos

- A orientação a objetos ajuda a lidar com a mudança
 - os objetos, e conceitos como abstração, encapsulação, herança e polimorfismo, suportam **extensibilidade** e **modificabilidade**
- É mais do que um paradigma de programação, pois inclui
 - **métodos de análise e desenho** de código (*code level design*)



<https://jorgeps14.medium.com/converting-classes-to-database-tables-12a70ebfb794>



Orientação a Objetos: um pouco de história

- O Simula 67 foi a primeira linguagem de programação orientada a objetos
- Foi desenvolvido
 - por Kristin Nygaard e Ole-Johan Dahl, na Noruega
 - para suportar a simulação discreta com eventos, uma técnica da investigação operacional, usada por exemplo para análise de tráfego
 - a extensibilidade e a reutilização eram dois atributos de qualidade muito importantes nos simuladores a desenvolver



Organização da Disciplina

Objetivos

Familiarizar os alunos com os conceitos e a prática da

análise, desenho e programação
orientadas a objetos

de forma a que se tornem capazes de

- analisar os requisitos de aplicações de alguma dimensão
- desenhar e implementar software que cumpra não só os requisitos funcionais como seja também **robusto**, fácil de **compreender** e suporte facilmente a **evolução**.

Abordagem

- Conceitos chave da programação orientada a objetos
 - como suportam importantes objetivos do desenho como a flexibilidade, extensibilidade, modificabilidade, reutilização, robustez
- Processo de análise e desenho
- Padrões de desenho que podem aplicar
 - padrões que codificam soluções bem estabelecidas para problemas comuns
- Princípios para escolher entre alternativas
 - princípios que descrevem boas práticas
- Técnicas para documentar e comunicar numa notação *standard* o resultado da análise e do desenho

Objetivos de Aprendizagem

No final do semestre, é esperado que:

- seja fluente na utilização de notações standard de análise e desenho OO
- saiba usar estas notações de forma efetiva na formulação de problemas e exploração de soluções
- perceba os conceitos chaves da programação OO e saiba implementar em Java de forma fidedigna soluções de desenho expressas nestas notações;
- esteja familiarizado com um conjunto de padrões standard de desenho e de implementação OO.

Tópicos

- **Programação Centrada em Objetos:** Noções chave da programação centrada em objetos e suporte destas noções na linguagem Java: composição, herança, redefinição, sobrecarga, polimorfismo, ligação dinâmica, abstração, contratos, subtipagem comportamental. Encapsulação e controlo de acesso. Modularização e suporte dado por packages e módulos. Reflexão e carregamento dinâmico de classes. Suporte à reutilização dado por genéricos, bibliotecas e frameworks.
- **Análise e Desenho Centrado em Objetos:** Os produtos da análise de requisitos. A análise OO: modelo de domínio e modelo de casos de uso. Desenho de sistemas de classes centrado na atribuição de responsabilidades: padrões de desenho para atribuição de responsabilidades, realização de casos de uso, modelo de classes. Vários padrões de desenho e princípios que contribuem para soluções mais preparadas para a mudança, para a divisão do trabalho, para a reutilização e para a robustez. A representação de soluções de desenho recorrendo a uma notação standard, o UML.

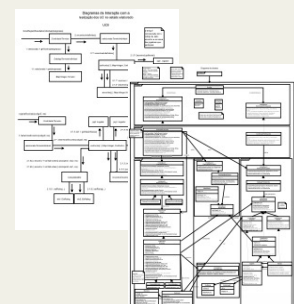
Avaliação

- **Trabalhos Práticos 25%**
 - 2 trabalhos realizados em grupo de 2 elementos
 - para cada trabalho há uma prova de aferição individual realizada na aula teórica a seguir à data de entrega (~20 mn)
- **Quizzes 7,5% ou 0%**
 - ~12 quizzes realizados nas aulas teóricas
 - média das notas, sem as 2 piores
- **Exame 67,5% ou 75%**
 - 1 prova individual escrita de 3h, sem consulta
 - **Nota mínima:** 9 valores

Trabalhos Práticos

- **Trabalho 1 programação 15%**
 - para concretizar em 3-4 semanas
 - entrega a 10 abril no servidor git dos alunos
 - nota individual: em função da prova de aferição, realizada na aula teórica de 11 de abril
- **Trabalho 2 análise e desenho 10%**
 - para concretizar em 2-3 semanas
 - entrega a 1 de junho no moodle
 - nota individual: em função da prova de aferição, na aula teórica de 2 de junho

Lines of Code	2,563
Lines	4,808
Statements	1,162
Functions	302
Classes	57
Files	55
Comment Lines	981
Comments (%)	27.7%



Desmascarando alguns Mitos sobre a Avaliação

“Os projetos que são extensos e trabalhosos só valem 25% da nota total”

O objetivo principal **não é avaliar os alunos** mas promover a **aquisição de conhecimentos**

“A percentagem do exame na avaliação é exagerada, quando comparada com os projetos desenvolvidos.”

É o exame que serve para **avaliar** o grau em que cada aluno atingiu os **objetivos de aprendizagem**

Racional do Método de Avaliação


Trabalhos Práticos

- O objetivo principal **não é avaliar os alunos**
- Mas promover a **aquisição de conhecimentos**
 - que são difíceis de aprender sem “meter as mãos na massa”
 - que exigem um contexto que caia dentro do *programming-in-the-large* e, que consequentemente, têm de ter alguma dimensão
- A parte mais importante não é o que entregam (não são produtos!) mas o que aprenderam ao fazê-lo
- O tempo que passam de volta de um trabalho tem um **valor educacional**

Exame

- É o que realmente serve para **avaliar** o grau em que cada aluno atingiu os **objetivos de aprendizagem** da disciplina

Exame e Objetivos de Aprendizagem



Desenvolvimento Centrado em Objetos

Licenciatura em Engenharia Informática

EXAME Tipo
2022/2023

Duração 3h 20 valores

Suponha que se pretende desenvolver uma aplicação de Q&A. O sistema permite que um utilizador autenticado contribua com questões. Um utilizador pode ver as respostas que já existem para as várias questões, votar positiva ou negativamente (upvote/downvote) nestas respostas, ou contribuir com uma resposta da sua autoria. Foi identificado também que o sistema teria o caso de uso opagar resposta. Neste caso de uso o utilizador pede ao sistema para listar as respostas da sua autoria, depois indica a resposta que pretende remover, através de um identificador único da resposta, e o sistema confirma que essa resposta foi removida com sucesso.

GRUPO I [4]

1. Apresente um modelo do domínio.
2. Considere o cenário de sucesso do caso de uso opagar resposta.
 - (a) Desenhe o SSD deste cenário.
 - (b) Escreva as pré e pós-condições do contrato da 2ª operação.
3. Das afirmações seguintes, indique as afirmações que são falsas, justificando.
 - (a) A atividade de análise é focada no problema e nos seus requisitos.
 - (b) A identificação dos casos de uso faz parte da análise orientada a objetos (OO).
 - (c) Da atividade de análise OO deve resultar um conjunto de classes de software.

GRUPO II [5]

1. Pretende-se agora desenhar uma realização do caso de uso opagar resposta.
 - (a) Desenhe um diagrama de interação para a 1ª operação do caso de uso, assumindo que essa operação fornece uma lista com as respostas da autoria do utilizador autenticado.
 - (b) Mostre, com um diagrama de classes, as consequências das decisões tomadas (que classes de software existem e que métodos e atributos têm).
2. As decisões que tomou aderem ao princípio do low representation gap? Indique pelo menos duas vantagens de aderir a este princípio.
3. Das afirmações seguintes, indique as afirmações que são falsas, justificando.
 - (a) Da atividade de desenho OO deve resultar um conjunto de classes de software a programar.
 - (b) A identificação dos métodos e atributos das classes que fazem parte da solução é feita durante a fase de implementação.
 - (c) A atividade de desenho OO foca-se tanto no desenho de todos os elementos da solução, desde a interface do sistema até ao armazenamento dos dados que precisam de ser guardados de forma persistente.

GRUPO III [8]

Ainda no contexto da aplicação Q&A, foi identificado um conjunto de restrições a que as votações das respostas estão sujeitas. Em primeiro lugar, um utilizador só pode votar nas respostas dos outros; além do sentido do voto é registada a data. O utilizador pode posteriormente mudar o sentido do seu voto numa resposta desde que esta ainda não tenha sido apagada. Os utilizadores ganham pontos positivos/negativos para o seu *karma* quando as suas contribuições recebem votos positivos/negativos.

1. Suponha que foi decidido ter duas classes, *UpVote* e *DownVote*, para representar os dois tipos de votos e que os construtores de ambas as classes recebem um objeto *User* e um objeto *Answer*, correspondendo ao votante e à resposta a ser votada.
 - (a) Escreva o contrato de um dos construtores, assumindo que a classe *Answer* tem um método para saber o autor da resposta.
 - (b) O que acontece a esse contrato se a classe *User* não redefinir nenhum método herdado de *Object*? Indique as assinaturas dos métodos de *User* que precisam de ser redefinidos.
 - (c) Esboce uma solução em que o que há de comum entre os dois tipos de votos está fatorizado e mostre como essa solução são implementados os construtores de *UpVote* e *DownVote*.
 - (d) Indique qual o efeito de declarar estas duas classes como *final*.
2. Suponha que a classe *QAController* tem um atributo do tipo *User* que corresponde ao utilizador autenticado e um método para registar um voto positivo/negativo desse utilizador numa resposta.
 - (a) Esboce uma solução preparada para a mudança do algoritmo usado para determinar o efeito que um voto positivo/negativo tem no *karma* do autor da resposta votada.
 - (b) Suponha que se decidiu implementar diferentes algoritmos e permitir que a escolha do algoritmo a usar seja configurável, recorrendo a um ficheiro de configuração. Indique os elementos que precisaria de juntar à solução anterior e o seu papel.
 - (c) Indique de que forma poderia tornar a sua solução robusta relativamente a situações em que o ficheiro de configuração não existe ou não tem uma configuração válida para o algoritmo.

GRUPO IV [3]

Considere o excerto da classe fornecida abaixo.

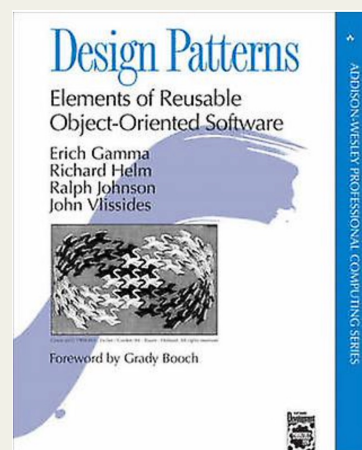
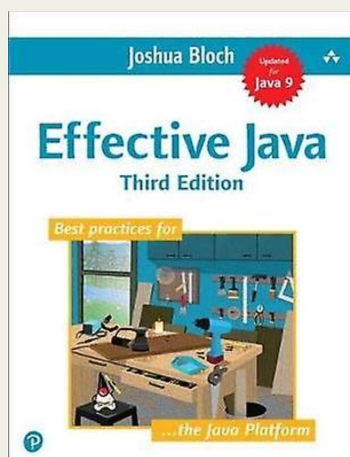
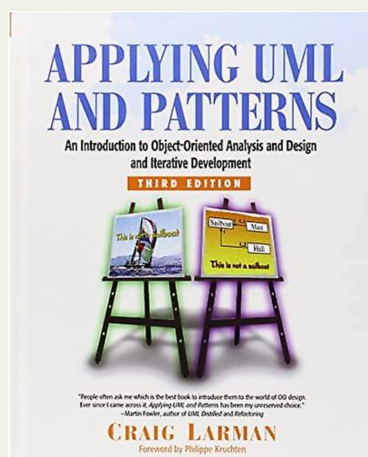
```
public class ServiceRegistry {
    private static ServiceRegistry instance = null;
    public static synchronized ServiceRegistry serviceRegistry() {
        if (instance == null) {
            instance = new ServiceRegistry();
            instance.registerDefaultServices();
        }
        return instance;
    }
    private final Map<Class<? extends Service>, Service> services = new HashMap<>();
    private void registerDefaultServices() {
        addService(new ServiceA());
    }
    ...
}
```

1. Que padrões e idiomas reconhece terem sido aplicados?
2. A inclusão de *synchronized* na declaração de *serviceRegistry()* que garantias dá?
3. Implemente o método *addService* que permite adicionar um novo objeto *Service* ao registo.
4. Indique, justificando, se poderíamos declarar *services* com o tipo *Map<Class<Service>,Service>*.

Quizzes

- Pequenos questionários realizados nas aulas teóricas sobre a matéria que está a ser lecionada
 - Online
 - Ou em papel, se necessário, mas implica que me avisem atempadamente
- São opcionais
 - Quem preferir pode ser avaliado apenas através dos trabalhos práticos e exame, sendo que neste caso o exame vale 75%
 - Basta **não responder a nenhum quiz**
- São um incentivo para que façam uma aprendizagem continuada e possibilitam feedback (quase) imediato

Bibliografia



Leituras Recomendadas. Junto aos sumários das aulas de cada semana serão indicadas as leituras recomendadas e, sempre que se justifique, indicados outros recursos considerados relevantes.

DCO semana-a-semana



Semana 1 (17/2)

T1 Apresentação da disciplina: objetivos, modo de funcionamento e método de avaliação. A disciplina em contexto: desafios do *programming-in-the-large* e a orientação aos objetos.

T2 Classes e Interfaces em revisão. Representação com UML de decisões relativas à estrutura e ao comportamento através de diagramas de classes e diagramas de interação. Polimorfismo de subtipos e o *dynamic dispatching*.

TP1 Apoio à instalação e utilização do controlo de versões com o git.

Leituras Recomendadas:

- Capítulos 1 do livro [Larman2004]

Dicas da semana:

- Fazer o guião git.
- Se precisarem de apoio na instalação ou na realização do guião e usam um portátil para trabalhar, levem-no para a aula TP.
- Na aula TP2, vamos ver os exercícios da série 1. Devem resolver antes da aula, pelo menos, o exercício 1.



T 1 DCO Apresentação

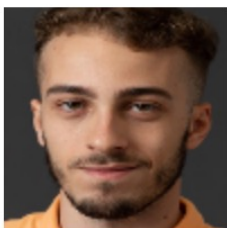
Corpo docente



Antónia Lopes

Responsável

Turnos: T21, TP22, TP24



Diogo João Ribeiro Branco

Turnos: TP21, TP23, TP25, TP26

Receita para isto correr bem

- Assistam às **aulas** teóricas e teórico-práticas de uma forma empenhada e participativa
- Acompanhem a matéria através do **material bibliográfico** aconselhado
- Realizem os **exercícios e trabalhos práticos** propostos
- Discutam com os docentes e colegas as dúvidas que tenham no decurso da realização dos mesmos
- Usem o **fórum do moodle** para colocar as vossas questões ou discutir assuntos de interesse da disciplina
- Apareçam durante os horários de esclarecimento de **dúvidas** do corpo docente sempre que tiverem dúvidas ou dificuldades