PROGRAMAÇÃO ORIENTADA AOS OBJECTOS EM JAVA6

Prof. F. Mário Martins

fmm@di.uminho.pt

Departamento de Informática
Universidade do Minho

2008/2009

Departamento de Informática / Escola de Engenharia
Universidade do Minho

Ano Lectivo 2008/2009 (2º semestre)

Escolaridade

2 x 1h T + 1 x 1 TP + 1 x 2h P

Cursos a que é leccionada:

Curso	Ano	Código
Licenciatura em Ciências da Computação	*2º*	

Responsável:

Prof. F. Mário Martins

Equipa Docente:

Docente	T	TP	P
Prof. F. Mário Martins	*2 (2 x 1h)*		*2 (1 x 2h)*
Prof. António Nestor Ribeiro		*2 (2 x 1h)*	*2 (1 x 2h)*

FUNCIONAMENTO:

Aulas Teóricas onde são introduzidos os conceitos fundamentais deste paradigma tão importante e tão particular, em articulação com as aulas Teórico-Práticas e Práticas que têm por objectivo consolidar tais noções à custa quer de métodos de análise de projectos já realizados, quer através da exemplificação prática da sua utilização (ou até das consequências da sua falta de utilização) em programas escritos utilizando **JAVA6** como linguagem de programação.

Estrutura e Objectivos Pedagógicos:

As aulas teóricas terão por objectivo a exposição da matéria fundamental que permita caracterizar o *paradigma da programação orientada aos objectos* bem como as construções da linguagem JAVA6 que implementam as construções fundamentais do paradigma.
Esta disciplina pretende realizar a transição entre a *programação em pequena escala* até aqui realizada nas disciplinas anteriores, onde se estudaram estruturas de dados mas onde as mesmas raramente foram aplicadas em aplicações de envergadura real, introduzindo-se agora todos os problemas inerentes ao desenvolvimento de aplicações de carácter e envergadura mais reais.
Surgem assim, conceptual e pragmaticamente, questões tais como a modularidade conceptual e de implementação, questões relacionadas com a correcção de erros e eficiência, e, ainda, com a própria facilidade de manutenção e extensão das aplicações desenvolvidas. Procura-se mostrar como as metodologias *orientadas aos objectos* permitem satisfazer um elevado e importante número de princípios cada vez mais fundamentais da Engenharia de Software.
Pretende-se atingir uma boa compreensão da evolução histórica do paradigma, surgido como muitos outros nos anos 70, mas que só mais recentemente atingiu uma enorme projecção fora das universidades, das suas principais características identificadoras e dos conceitos formais que o consubstanciam e distinguem de outros.
A título de exemplo, e procurando introduzir certas novas noções a partir de outras noções introduzidas em anteriores disciplinas, o conceito de *objecto*, unidade modular fundamental ao paradigma, é associada às noções de módulo, abstracção de dados, protecção e encapsulamento tão importantes para a independência do software.
Pretende-se, a este nível, a completa compreensão teórica de conceitos tais como *objecto*, *classe*, *polimorfismo*, *hierarquia*, *herança*, *agregação*, etc. Porém, sendo estes também os conceitos fundamentais à compreensão não só da tecnologia mas também dos métodos de análise e concepção de software baseados no paradigma OO, em termos pragmáticos, os mesmos deverão ser profundamente apresentados (segundo diferentes perspectivas da sua implementação até) e no final da disciplina perfeitamente assimilados.
Apresentar-se-ão igualmente alguns dos principais problemas típicos que se colocam na concepção de aplicações orientadas aos objectos, em particular os que têm a ver com decisões de classificação de novas classes a desenvolver. Aqui, procura-se uma clara compreensão da distinção entre os dois principais tipos de *herança* (lógica ou física, simples ou múltipla) e ainda a distinção clara entre os

mecanismos de *agregação* e de *herança*. Ao longo da disciplina procurar-se-á sempre, ao nível das aulas teóricas, apresentar comparações entre as mais diversas maneiras como a actual tecnologia baseada no paradigma OO implementa tais conceitos fundamentais. Apresentam-se assim as implementações realizadas em algumas linguagens de objectos mais usadas, em particular comparando a linguagem usada na disciplina, JAVA, com C#.

No curso, a preocupação centrar-se-á na construção da camada computacional de aplicações orientadas aos objectos em isolamento da camada interactiva. As camadas interactivas a desenvolver serão básicas, ou seja, sem recurso a bibliotecas gráficas tais como AWT ou SWING, que implicariam, só por si, uma outra disciplina.

As aulas teórico-práticas consistirão, desde o seu início, da apresentação sintética e estudo, sob a forma de resolução de pequenos problemas adequados, da linguagem JAVA, em particular das regras a cumprir para realizar POO usando JAVA.

As aulas teórico-práticas serão também, quando conveniente, aulas de apresentação e análise de projectos, servindo de ponte entre a teoria e a prática de projecto. Serão aulas de transposição da matéria teórica para o desenvolvimento de projecto, visando reflexão sobre requisitos, analisar questões relacionadas com a concepção das classes de projecto, e a introdução de técnicas de escrita de código visando modularidade, encapsulamento e composição.

As aulas práticas são laboratoriais e presenciais. Estas aulas são realizadas em laboratórios e têm sempre componente prática em computador. Será usado o ambiente de desenvolvimento integrado (IDE) BlueJ como sistema de apoio ao desenvolvimento e prototipagem das pequenas aplicações que forem sendo desenvolvidas em JAVA, com a vantagem adicional da visualização de diagramas de classes e criação instantânea de instâncias e sua utilização e teste (prototipagem).

Numa fase posterior, e admitindo um razoável conhecimento por parte dos alunos das principais classes e métodos, bem como de alguns conceitos introduzidos nas aulas teóricas, as aulas práticas procurarão exercitar questões relacionadas com a concepção de maiores aplicações, nas quais o domínio dos mecanismos de herança e de agregação, de classificação, e de metodologia de concepção se torna importante.

A utilização do *mecanismo de excepções* de JAVA permitirá chamar a atenção dos alunos para os tão importantes aspectos da segurança e robustez do software, e, ao usar JAVA, tirar todo o potencial do muito claro e limpo mecanismo de tratamento de excepções.

São palavras e ideias chave da engenharia do software que importa reforçar como bem abordadas por este paradigma: a *modularidade* e o *encapsulamento* (pelo uso de objectos, classes e mensagens), a *flexibilidade* (via polimorfismo), a *classificação* (via hierarquia), a *reutilização* (via mecanismos de herança e agregação), a *extensibilidade* e a *generalidade* (via *polimorfismo natural* e *polimorfismo paramétrico* – classe genéricas desde Java5), entre outras.

Deve ainda ser realçada a *verticalidade* da utilização destes conceitos no projecto de software, dado que os mesmos podem ser aplicados desde a análise à implementação de sistemas seguindo o paradigma OO, por exemplo, noutras disciplinas do curso.

Síntese de Objectivos:

A disciplina tem por objectivo principal completar a formação dos alunos na área da programação, pela introdução de outros modelos de programação existentes com grande capacidade na resolução de classes particulares de problemas. É objectivo da disciplina a apresentação do *Paradigma da Programação Orientada aos Objectos*, nas suas bases teóricas, das suas capacidades específicas e das áreas da sua particular aplicação.
É particularmente importante que esta disciplina e este paradigma estabeleçam, ao nível do curso, a distinção entre a *programação em pequena escala* e os problemas da *programação em grande escala*, designadamente a adopção de técnicas de concepção e desenvolvimento modulares e escaláveis, em particular explorando todas as que o paradigma da programação por objectos oferece, neste caso particular usando as características da linguagem JAVA.
Compreendidas as potencialidades do paradigma para a resolução de certas categorias de problemas, pretende-se, do ponto de vista prático, que os alunos se tornem auto-suficientes na escrita de aplicações em JAVA, usem em seu favor o IDE BlueJ, até para documentação rápida de projecto, e adquiram o conhecimento genérico suficiente para que, posteriormente, e por si, possam, se tal for necessário, desenvolver as suas capacidades de utilização da linguagem e dos conceitos em disciplinas mais avançadas do curso, tais como Desenvolvimento de Sistemas Software, Técnicas Avançadas de Orientação aos Objectos, Sistemas Multimédia e Gráficos, Sistemas Operativos, Criptografia, etc.

Sistema de Avaliação:

A avaliação tem uma componente teórica (1 teste teórico), a realizar em data já anunciada, e uma componente prática, ambas obrigatórias. Tal significa que um aluno que não obtenha a classificação mínima fixada para cada componente não será aprovado.
A **componente prática** consistirá da realização de **1 trabalho prático**, sob a forma de trabalho de grupo de no máximo 3 elementos. A não realização do trabalho implica de imediato a reprovação do aluno à disciplina, passando a ser **não admitido** a exame, e considerado, para efeitos estatísticos, como **não avaliado**.
Para a componente prática a nota mínima deverá ser de **10 valores**, caso contrário o aluno é igualmente **não admitido**, ainda que seja estatisticamente considerado como **avaliado**.
Nas aulas laboratoriais da disciplina são propostos e acompanhados pequenos exercícios, agrupados em fichas laboratoriais da disciplina. Tais trabalhos pretendem servir de guião à componente prática da cadeira, e fios condutores do estudo dos alunos, tendo ainda como objectivo serem auxiliares à resolução do trabalho final. Tais trabalhos servirão também de base para muitas das questões que serão colocadas aos alunos na avaliação da sua formação teórica.
A **nota teórica** será obtida através da realização de **1 teste individual escrito**, sendo a nota mínima necessária para a realização da componente teórica **9.5 valores**. Caso o aluno obtenha uma classificação entre 9 e 9.4, esta nota teórica, devidamente pesada, é considerada para efeitos de

média com a nota prática, sendo-lhe no entanto descontado 1 valor na sua média final, devendo esta ser obviamente positiva após tal ajuste (>= 9.5).
O aluno poderá obter a sua nota teórica em exame de recurso, desde que possua já nota prática.
A **nota final** da disciplina será encontrada, após satisfação das regras anteriores, e, salvo o caso especial de nota teórica entre 9 e 9,4 anteriormente referido, pela aplicação da seguinte fórmula simples:

Nota Final = (Nota Teórica x 0, 55) + (Nota Prática x 0, 45)

O trabalho prático entregue pelos alunos de um dado grupo terá uma classificação que poderá ser, quando tal se justifique, individualizada. A não presença, injustificada, de um dos elementos de um dado grupo na apresentação e discussão do respectivo trabalho implica a sua não avaliação e consequente reprovação.

A classificação final do trabalho prático entregue, deverá ser calculada em função da seguinte escala de critérios e valores:

Escalão	Nota
Sem qualidade	6
Pouca qualidade	8/9
Qualidade mínima	10
Qualidade média	13
Bom trabalho	15
Muito bom trabalho	17
Trabalho excelente	18-20

A avaliação dos trabalhos terá em consideração diversas componentes, tais como:

- *qualidade e complexidade das decisões de projecto;*
- *qualidade, em apresentação e síntese, do relatório apresentado;*
- *qualidade do código fonte apresentado;*
- *qualidade da execução (com ou sem erros, satisfaz ou não requisitos, etc.);*
- *qualidade da apresentação ao utilizador (interface), caso tal se aplique;*
- *facilidade de utilização do programa sem ler manuais;*
- *nível da prestação oral dos elementos do grupo;*

Sistema de Melhoria de Notas:

Implica a satisfação de todos os requisitos anteriores, excepto, obviamente, a necessidade de uma frequência às aulas.

PROGRAMAÇÃO ORIENTADA AOS OBJECTOS

CONTEÚDO PROGRAMÁTICO DETALHADO

PROGRAMAÇÃO ORIENTADA AOS OBJECTOS EM JAVA6

1. MATÉRIA TEÓRICA DE PPO:

1.1.- Introdução à Programação por Objectos.

- Origem do paradigma. Via Simulação. Via Computação.
- Conceitos básicos fundamentais.
- Modelos: de processos versus de objectos.
- A procura da modularidade no software.
- Independência do contexto como condição fundamental.
- Encapsulamento versus independência e modularidade.
- Modularização pelos dados: a solução em PPO.

1.2.- Noção de "Objecto" em PPO.

- Noção de "objecto" em PPO. Estrutura e Comportamento.
- Encapsulamento e protecção nos objectos.
- Interacção entre objectos. Mensagens vs. Métodos.
- Introdução ao Polimorfismo.
- Tipos de objectos: instâncias e classes.

1.3.- Classes, Hierarquia de Classes e Herança.

- Definição de Classe em PPO.
- Relação Classe-Instâncias. Introdução.
- Mecanismo de instanciação. Construtores.
- Classes e sua Hierarquia. Superclassificação.
- Relações entre Classes. A herança.
- Herança como mecanismo de reutilização e de programação incremental.
- Herança simples e múltipla.
- Algoritmo de procura de métodos.
- Herança versus Agregação.

1.4.- Classes e Herança.

- Criação de Classes.
- Classes "run-time" versus Classes para "compile-time".

- Tipos estáticos e dinâmicos das variáveis.
- Polimorfismo; “static” e “dynamic binding”.
- Classes não instanciáveis.

1.5.- Classes Abstractas.

- Definição de Classe Abstracta. Importância das Classes Abstractas.
- Classes Abstractas vistas como Tipos Abstractos de Dados.
- Classes Abstractas como mecanismo de abstracção.
- Classes Abstractas como mecanismo de reutilização e de extensibilidade.
- Polimorfismo. Estudo dos diferentes tipos.

1.6.- Estudo particular das colecções parametrizadas de JAVA5/6.

- Interfaces: Collection<E>, List<E>, Set<E>, Map<K,V>.
- Estudo dos iteradores sobre colecções: for e Iterator<E>.
- Tipos enumerados.

1.7.- Concepção de aplicações em PPO.

- Subclassificação e herança versus agregação.
- Subclasses como especializações.
- Classes abstractas versus interfaces.
- Subclasses para implementação.
- Algumas regras de concepção em PPO.

2. PROGRAMAÇÃO POR OBJECTOS EM JAVA: ESTUDO DA LINGUAGEM JAVA (JAVA 6.4)

2.1.- Programação por Objectos em JAVA.
Características do ambiente de desenvolvimento JDK6.
A JVM ("Java Virtual Machine"). Byte-code.
Estrutura dos programas.
Bibliotecas. Packages.
Introdução ao IDE BlueJ. Características e funcionalidade.

2.2.- Tipos básicos (não objectos) e operadores.
Numéricos. Boleanos. Declarações.
Arrays de Java: suas vantagens e inconvenientes.

2.3.- Estruturas de controlo.
Condicionais simples e compostas.
Estruturas Iterativas.

2.4.- Definição de Classes e Instâncias em JAVA.
Construtores. Métodos e variáveis de instância e de classe.
Tipos de qualificadores de visibilidade e acesso das variáveis e constantes.

2.5.- Hierarquia de Classes em JAVA.
Herança simples. Redefinição e sobreposição de métodos e variáveis.
Classes e subclasses. Exemplos clássicos.
Classe Object. Classes versus Packages.
Compatibilidades entre instâncias de classes e subclasses.
O mecanismo de "dynamic type checking".
Métodos "standard" equals(), clone() e toString().

2.6.- Classes Abstractas em JAVA.
Declaração.
Polimorfismo e sua utilização. Regras da linguagem.
"Static-checking" vs. "Run-time-checking" em JAVA.
Exemplos com classes abstractas.

2.7.- O mecanismo de Excepções da linguagem JAVA.
Cláusulas *try, catch, finally, throws e throw*.
Regras de utilização.

2.8.- *Interfaces* JAVA como especificações de Tipos de Dados.
Classes como subclasses e classes como subtipos. Distinção.
Herança múltipla de Interfaces em JAVA.
Regras para a implementação de Interfaces em Classes.

2.9.- Tipos parametrizados e Colecções de JAVA6.
Estudo da JCF *("Java Collections Framework"*).
Os três tipos de colecções de JAVA: List<E>, Set<E> e Map<K,V>.
Implementações de List<E>: ArrayList<E>: for(each) e iteradores;
O problema do clone() de colecções.

Colecções não-covariantes e utilização de “wildcards”.
Implementações de Set<E>: HashSet<E> e TreeSet<E>;
Boxing e Unboxing automáticos.
Implementações de Map<K,V>: HashMap<K,V>, TreeSet<K, V>;
Implementação da interface Comparator<E>.
Tipos Enumerados.

2.10.- Estudo das *Streams* de JAVA.
Streams de caracteres versus streams de bytes.
Streams de input e streams de output. As classes abstractas Writer e Reader.
Subclasses de Writer e Reader. Mecanismo de “aninhamento” de streams.
As ObjectStreams como mecanismo de persistência de dados. A
Interface Serializable.
Exemplos de eficiência no uso de streams. Exemplos de teste.
Medida de eficiência. Comparação da eficiência das diversas soluções.

2.11.- Classes Genéricas de JAVA.
Importância das “inner classes” e de outras construções de JAVA como suporte efectivo à implementação de classes genéricas .

BIBLIOGRAFIA

SOBRE O PARADIGMA DA PROGRAMAÇÃO POR OBJECTOS USANDO A LINGUAGEM JAVA6

JAVA5 e Programação por Objectos
F. Mário Martins, Editora FCA, Série Tecnologias de Informação,
ISBN-972-722-548-9, Setembro de 2006, 2ª. Edição

% Um livro que apresenta as características fundamentais do paradigma da PPO e como tais características podem e devem ser implementadas usando, por exemplo, JAVA, para a criação metodológica de aplicações %

Programação Orientada aos Objectos em JAVA5
F. Mário Martins, Notas Pedagógicas, revisão de 2008.

% O conjunto dos apontamentos teóricos da disciplina, tal como apresentados nas aulas teóricas, aos quais se anexam diversos exemplos concretos de pequenos projectos. %

Laboratórios de JAVA6: Fichas Práticas
F. Mário Martins, Março de 2007 (Revisão de 2008)

% O conjunto dos apontamentos laboratoriais da disciplina, tal como apresentados nas aulas práticas, com síntese teórica da matéria e respectivas resoluções. %

SOBRE A LINGUAGEM JAVA5

JAVA 1.5 Tiger: A Developer´s Notebook
Brett McLaughlin and David Flanagan, O´Reilly, 2004

Java in a Nutshell
D. Flanagan, O´Reilly & Associates, 5th Edition, 2004

% A 5ª edição do livro de apresenta todas as bibliotecas de JAVA5: classes e suas API. %

The JAVA™ Programming Language, Sun Microsystems, July, 2004.
Ken Arnold, James Gosling, David Holmes

SOBRE BLUEJ

BlueJ Manual On-Line, M. Kolling, em www.bluej.org
"Objects First with Java", D. Barnes e M. Kolling, Prentice Hall-PersonEducation, 2003.

Manuais ON-LINE, APIs, Fontes, etc. de JAVA jsdk6

Página da disciplina: http://sim.di.uminho.pt/*ensino.php3*
Ver ainda: http://www.java.sun.com
http://www.bluej.org

AVALIAÇÃO – SÍNTESE:

- **AVALIAÇÃO PRÁTICA: 1 TRABALHO DE GRUPO (máx. 3)**

  NOTA >= 10 => APROVADO PARTE PRÁTICA
  NOTA < 10 => REPROVADO NA DISCIPLINA (S/ RECURSO)

- **AVALIAÇÃO TEÓRICA: 1 teste teórico (a 23/Junho/2009)**

  NOTA >= 9,5 => APROVADO

  NOTA FINAL = TEÓRICA * 0,55 + PRÁTICA * 0,45

  NOTA < 9 => REPROVADO (não fazemos orais)
  => EXAME DE RECURSO

  NOTA entre 9 e 9.4 =>
  NOTA FINAL = (TEÓRICA * 0,55 + PRÁTICA * 0,45) - 1
  Se NOTA FINAL >= 9,5 => APROVADO com NOTA FINAL
  senão REPROVADO

NOTAS DE FUNCIONAMENTO IMPORTANTES:

- **INSCRIÇÕES NAS TURMAS PRÁTICAS SEM RESTRIÇÕES;**
- **SERÁ REALIZADO APENAS 1 TRABALHO PRÁTICO DE GRUPO (no máximo 3 alunos por grupo), CUJO ENUNCIADO SERÁ TORNADO PÚBLICO EM FINAL DE MARÇO;**
- **A ENTREGA PRESENCIAL DO TRABALHO REALIZAR-SE-Á NA SEMANA DE 8 de JUNHO, devendo o mesmo ser entregue por via electrónica em data e endereço a divulgar posteriormente;**
- **MELHORIAS DE NOTAS IMPLICAM A REALIZAÇÃO DO TRABALHO PRÁTICO E DO TESTE/EXAME TEÓRICO;**
- **NÃO HÁ, PORTANTO, NOTAS PRÁTICAS CONGELADAS.**
- **FAZER CONSULTAS REGULARES a sim.di.uminho.pt -> Ensino**

PLANEAMENTO SEMANAL

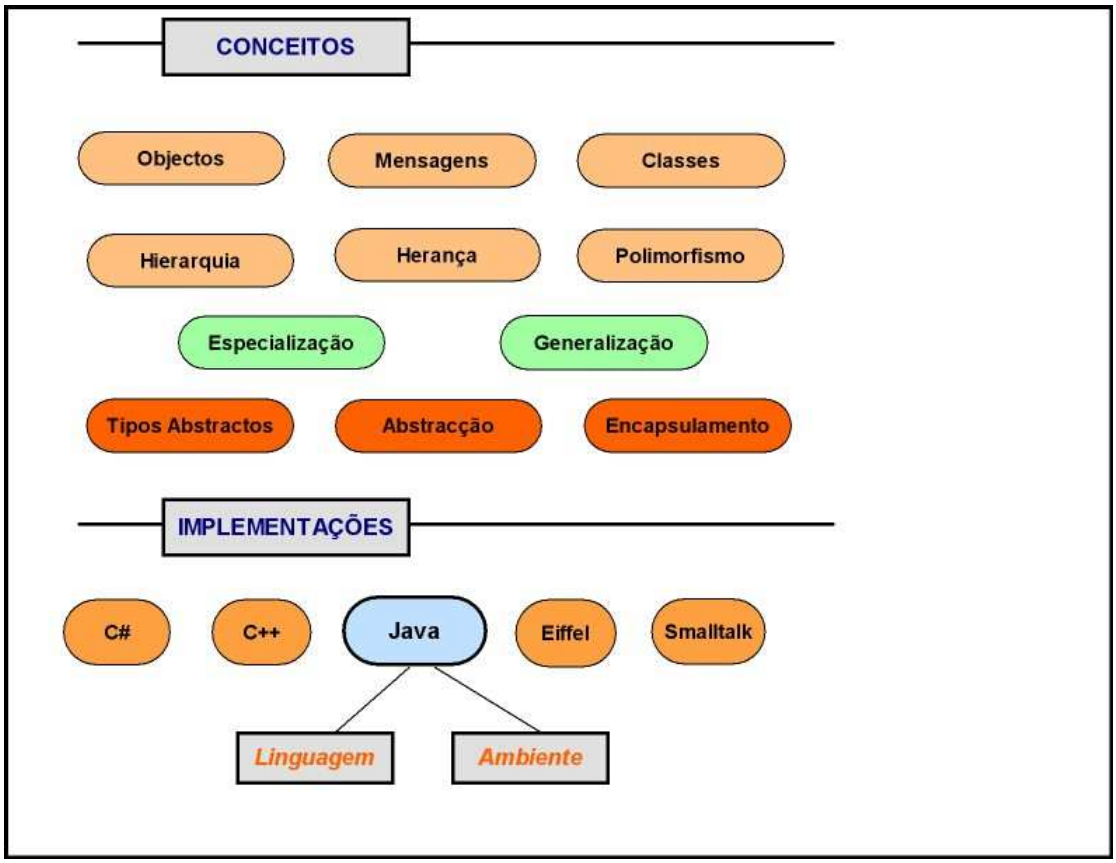
Programação Orientada aos Objectos

2º ano/2º semestre - LCC
2008-2009
Aulas Teóricas

	Data	Sumário da Aula
1	02-03-2009	*Apresentação da disciplina: Funcionamento e Avaliação*
2	03-03-2009	*Introdução à Programação por Objectos; Encapsulamento;*
3	09-03-2009	*Encapsulamento - Continuação; Exemplos;*
4	10-03-2009	*Noção de Objecto; Estrutura e Comportamento;*
5	16-03-2009	*Mensagens como mecanismo de abstracção; o.m();*
6	17-03-2009	*Definição de Classe; Criação de Classes simples: Ponto2D*
7	23-03-2009	*Métodos standardizados; Criação de classes;*
8	24-03-2009	*Criação de Classes: Exemplos adicionais; Método clone();*
9	30-03-2009	*Criação de Classes por Agregação: Circulo(s)*
10	31-03-2009	*Continuação do estudo da agregação*
	06-04-2009	Páscoa
	07-04-2009	Páscoa
11	13-04-2009	*Colecções de JAVA5/6; Tipos Collection<E>: List<E>, Set<E> e Map<K,V>;*
12	14-04-2009	*Estudo de ArrayList<E>: implementação; Métodos addAll() e clone();*
13	20-04-2009	*Continuação; Iteradores: for e Iterator<E>;*
14	21-04-2009	*HashSet<E> e TreeSet<E>;*
15	27-04-2009	*Herança e Polimorfismo;*
16	25-04-2009	*Classes Abstractas e Polimorfismo: Exemplos*
17	04-05-2009	*Estudo de HashMap<K,V> e TreeMap<K,V>; Wildcards;*
18	05-05-2009	*Interfaces; Interfaces de Colecções; Comparator<T> e Serializable;*
	11-05-2009	Enterro da Gata
	12-05-2009	Enterro da Gata
19	18-05-2009	*Excepções: try & catch;*
20	19-05-2009	*Excepções (conclusão); Streams:*
21	25-05-2009	*Streams: FileReader; StringTokenizer; PrintWriter e ObjectStreams*
22	26-05-2009	*Wildcards: ? extends T e ?*
23	01-06-2009	*Conclusão da matéria*
24	02-06-2009	*Conclusão da matéria*
	08-06 a 12-06	Entrega presencial do Trabalho Prático
	23-06-2009	Teste Teórico de POO
	06-07 a 13-07	Exame de Recurso (data a definir pelo CC)

Docente: F. Mário Martins

SÍNTESE


CONCEITOS
Objectos
Mensagens
Classes
Hierarquia
Herança
Polimorfismo
Especialização
Generalização
Tipos Abstractos
Abstracção
Encapsulamento
IMPLEMENTAÇÕES
C#
C++
Java
Eiffel
Smalltalk
Linguagem
Ambiente