

Minha página principal ► Laboratório de Programação Orientada por Objectos ► Geral ► 2nd Mini-test

Data de início Quarta, 31 Maio 2017, 14:05

Estado Teste enviado

**Data de
submissão:** Quarta, 31 Maio 2017, 15:01

Tempo gasto 56 minutos 4 segundos

Nota **7,00** de um máximo de 20,00 (35%)

Pergunta 1

Incorreto

Pontuou -0,250 de 1,000

Destacar pergunta

No repositório local de Git, existem três "árvores" geridas pelo sistema de controlo de versões, são elas:

Selecione uma opção de resposta:

- ☒ a. Working directory, origin, remote. ✖
- ☐ b. Staging Area, Commit Area, HEAD.
- ☐ c. Working directory, Index (Stage), HEAD.
- ☐ d. Working directory, Master, HEAD.
- ☐ e. Não pretendo responder a esta questão.

A resposta correta é: Working directory, Index (Stage), HEAD.

Pergunta 2

Correto

Pontuou 1,000 de 1,000

Destacar pergunta

Relativamente aos princípios fundamentais de orientação por objetos, selecione a afirmação incorreta:

Selecione uma opção de resposta:

- ☐ a. Herança refere-se à possibilidade das subclasses herdarem propriedades das superclasses.
- ☒ b. Encapsulamento refere-se à possibilidade de definir classes públicas dentro outras classes. ✔
- ☐ c. Abstração refere-se à possibilidade de definir classes como abstrações para

conjuntos de objetos com propriedades similares.

- ☐ d. Polimorfismo refere-se à possibilidade da implementação de um método variar de acordo com a subclasse do objeto.
- ☐ e. Não pretendo responder a esta questão.

A resposta correta é: Encapsulamento refere-se à possibilidade de definir classes públicas dentro doutras classes.

Pergunta 3


Correto

Pontuou 1,000 de 1,000

 Destacar pergunta

Qual das seguintes **keywords** não é usada no tratamento de exceções em Java:

Selecione uma opção de resposta:


- ☐ a. **try**
- ☒ b. **final**

- ☐ c. Não pretendo responder a esta questão.
- ☐ d. **throw**
- ☐ e. **catch**

A resposta correta é: **final**

Pergunta 4

Correto

Pontuou 1,000 de 1,000


 Destacar pergunta

Atente ao seguinte código Java:

```
System.out.println("c" + (a ? "a" : "b"));
```

Indique qual das afirmações **não está** correta.

Selecione uma opção de resposta:


- ☐ a. Esta linha de código não tem erros de sintaxe.
- ☐ b. Se a variável **a** tiver o valor *false*, o programa imprime "cb" para a consola.
- ☐ c. A variavel **a** é do tipo *boolean*.
- ☒ d. A expressão não necessita tantos parêntesis. 
- ☐ e. Não pretendo responder a esta questão.

A resposta correta é: A expressão não necessita tantos parêntesis.

Pergunta 5

Incorreto

Pontuou -0,250 de 1,000

 Destacar pergunta

Relativamente a construtores e destrutores de classes em Java, indique qual das afirmações está incorreta.

Selecione uma opção de resposta:


- ☒ a. Como em Java, qualquer classe é automaticamente sub-classe de *Object*, pode-se usar a keyword **super** como forma de invocar o construtor de *Object*. ✖
- ☐ b. Caso se defina um destrutor numa classe, então será necessário invoca-lo explicitamente, senão o mecanismo de *garbage collection* irá ignorá-lo.
- ☐ c. Se uma classe **A** é sub-classe de **B**, e **B** apenas tem o construtor por omissão, então **A**, no seu construtor, não tem obrigatoriamente de invocar o construtor de **B**.
- ☐ d. Se uma classe **A** é sub-classe de **B**, e **B** tem apenas um construtor, tendo este parâmetros, então **A** terá obrigatoriamente, no seu construtor, de chamar o construtor de **B**, usando a keyword **super**.
- ☐ e. Não pretendo responder a esta questão.

A resposta correta é: Caso se defina um destrutor numa classe, então será necessário invoca-lo explicitamente, senão o mecanismo de *garbage collection* irá ignorá-lo.

Pergunta 6

Correto

Pontuou 1,000 de 1,000

 Destacar pergunta

Que coleção concreta de Java (classe de implementação) usaria para representar uma coleção ordenada de números em vírgula flutuante de precisão dupla, sem duplicados?

Selecione uma opção de resposta:

- ☐ a. **ArrayList<double>**
- ☒ b. **TreeSet<Double>**
✔
- ☐ c. Não pretendo responder a esta questão.
- ☐ d. **HashSet<double>**
- ☐ e. **TreeMap<Double>**

A resposta correta é: **TreeSet<Double>**

Pergunta 7

Incorreto

Pontuou 0,000 de 1,000

Destacar pergunta

Atente à seguinte linha de código em Java:

```
public <T extends K<? super T, ? extends T>> void weirdMethod(T weirdParam) {}
```

Indique qual das afirmações não é correta.

Selecione uma opção de resposta:

- ☐ a. A declaração do método está correta e o tipo T está bem definido, não dando qualquer erro de sintaxe.
- ☒ b. Não pretendo responder a esta questão. ✖
- ☐ c. **K** pode ser uma classe ou uma *interface*.
- ☐ d. O retorno deste método não poderá ser do tipo T, pois apenas se parametrizam os argumentos deste tipo de métodos.
- ☐ e. O metodo *weirdMethod* aceita como parâmetro um objeto do tipo **T**, que implementa a interface **K**, a qual é parametrizável com um qualquer objeto super-classe de T e outro que é sub-classe de T.

A resposta correta é: O retorno deste método não poderá ser do tipo T, pois apenas se parametrizam os argumentos deste tipo de métodos.

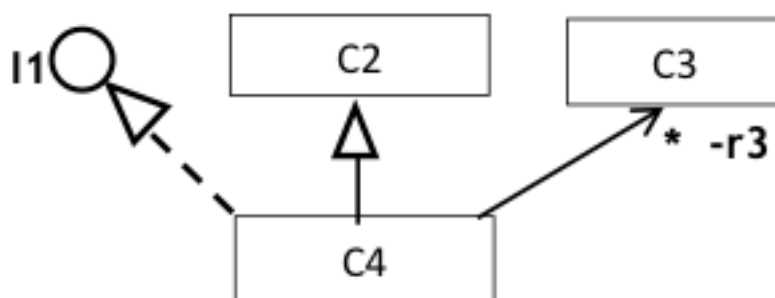
Pergunta 8

Correto

Pontuou 1,000 de 1,000

Destacar pergunta

Qual a tradução mais correta em Java da classe C4 do seguinte diagrama de classes UML?



Selecione uma opção de resposta:

- ☒ a. `class C4 implements I1 extends C2 { private HashSet<C3> r3; }`




- ☐ b. `class C4<I1> extends C2 { private Set<C3> r3; }`
- ☐ c. `class C4 implements I1 { private C2 c2; private Set<C3> r3; }`
- ☐ d. `class C4 extends I1, C2 { private HashSet<C3> r3; }`
- ☐ e. Não pretendo responder a esta questão.

A resposta correta é: `class C4 implements I1 extends C2 { private HashSet<C3> r3; }`

Pergunta 9


Correto

Pontuou 1,000 de 1,000

 Destacar pergunta

Relativamente aos diagramas de estados UML, selecione a afirmação incorreta:

Selecione uma opção de resposta:

- ☐ a. As transições são instantâneas.
- ☐ b. Podem-se utilizar estados compostos, contendo subestados sequenciais.
- ☐ c. Não pretendo responder a esta questão.
- ☐ d. A utilização de regiões ortogonais permite evitar a explosão combinatória de estados.
- ☒ e. Não podem existir múltiplas transições com o mesmo evento e o mesmo estado de origem. 

A resposta correta é: Não podem existir múltiplas transições com o mesmo evento e o mesmo estado de origem.

Pergunta 10

Correto

Pontuou 1,000 de 1,000

 Destacar pergunta

Em que consiste *refactoring*?

Selecione uma opção de resposta:

- ☐ a. Efetuar alterações a um programa para adicionar novas funcionalidades.
- ☐ b. Não pretendo responder a esta questão.
- ☐ c. Alterar o código de um programa para ficar coerente com um diagrama de classes.
- ☐ d. Efetuar alterações a um programa para corrigir *bugs* e garantir que passa nos

testes.

- ☒ e. Efetuar alterações à estrutura interna de um programa para o tornar mais fácil de compreender e modificar, sem alterar o comportamento observável do programa. ✓

A resposta correta é: Efetuar alterações à estrutura interna de um programa para o tornar mais fácil de compreender e modificar, sem alterar o comportamento observável do programa.

Pergunta 11

Incorreto

Pontuou -0,250 de 1,000

🚩 Destacar pergunta

Relativamente a testes unitários, indique qual das seguintes afirmações está incorreta.

Selecione uma opção de resposta:

- ☒ a. Testes "caixa-preta" pretendem verificar os requisitos/funcionalidades do sistema, enquanto os testes "caixa-branca" garantem a cobertura de código. ✗
- ☐ b. *Test-Driven Development (TDD)* é uma prática onde se implementa primeiro o teste que verifica uma funcionalidade, mesmo antes desta funcionalidade estar implementada.
- ☐ c. Os testes unitários não são tão eficazes se não se fizer *refactoring* a-priori.
- ☐ d. Normalmente a automação de testes unitários cobre funcionalidades de API, ao invés da GUI, por esta ser mais difícil de testar.
- ☐ e. Não pretendo responder a esta questão.

A resposta correta é: Os testes unitários não são tão eficazes se não se fizer *refactoring* a-priori.

Pergunta 12

Incorreto

Pontuou -0,250 de 1,000

🚩 Destacar pergunta

A ferramenta PIT permite executar "Mutation tests". Estes testes permitem:

Selecione uma opção de resposta:


- ☒ a. Identificar mutações que o código possa ter e corrigi-las. ✗
- ☐ b. Melhorar a qualidade do código, testando falhas introduzidas no código.
- ☐ c. Não pretendo responder a esta questão.
- ☐ d. Aumentar a cobertura de código, criando testes novos e diferentes dos já existentes.
- ☐ e. Introduzir falhas nos testes, e verificar de que forma o código reage.

A resposta correta é: Melhorar a qualidade do código, testando falhas introduzidas no código.

Pergunta 13

Incorreto

Pontuou 0,000 de 1,000

 Destacar pergunta

Para que servem os *Layout Managers* em SWING?

Selecione uma opção de resposta:


- ☐ a. Para efetuar a colocação e redimensionamento de componentes visuais dentro de um contentor, mediante um conjunto de restrições.
- ☐ b. Para gerir o sistema de janelas internas de uma aplicação, permitindo que haja sobreposição de janelas e garantindo sempre que as janelas são redesenhadas sempre que necessário.
- ☐ c. Para mostrar os componentes visuais da GUI de forma diferente consoante o sistema operativo onde a aplicação está a correr.
- ☐ d. Para gerir a visibilidade de componentes dentro de um contentor visual, nunca deixando que se oculte nenhum componente.
- ☒ e. Não pretendo responder a esta questão. ✖

A resposta correta é: Para efetuar a colocação e redimensionamento de componentes visuais dentro de um contentor, mediante um conjunto de restrições.

Pergunta 14

Incorreto

Pontuou -0,250 de 1,000

 Destacar pergunta

No SWING, para usar o método *paintComponent* para desenhar os nossos próprios gráficos, é necessário:

Selecione uma opção de resposta:


- ☒ a. Criar uma classe que seja sub-classe de *JFrame* e redefinir aí o método *paintComponent*. ✖
- ☐ b. Não pretendo responder a esta questão.
- ☐ c. Redefinir o método *paintComponent* em qualquer classe, desde que seja sub-classe de *JComponent*.
- ☐ d. Invocar o método *repaint()*, senão o método que se redefiniu nunca será invocado.
- ☐ e. Fazer *cast* do parâmetro *Graphics* para *Graphics2D*, tendo assim acesso às primitivas gráficas.

A resposta correta é: Redefinir o método *paintComponent* em qualquer classe, desde que seja sub-classe de *JComponent*.

Pergunta 15

Correto

Pontuou 1,000 de 1,000

 Destacar pergunta

O denominado "Open-Closed Principle" dos SOLID, pretende transmitir o seguinte:

Selecione uma opção de resposta:


- ☒ a. O código torna-se extensível, bastando acrescentar novas funcionalidades, sem ter de alterar código já existente. ✓
- ☐ b. O código está aberto a edição por parte de quem o pretende estender com novas funcionalidades, tendo apenas de o alterar em locais específicos.
- ☐ c. Não pretendo responder a esta questão.
- ☐ d. A extensibilidade do código atinge-se através do uso exclusivo de interfaces que outras classes referem.
- ☐ e. O código encontra-se fechado para modificação por parte de quem o pretende estender, sendo que para isso será sempre necessário pedir aos autores do código permissão para o fazer, pois apenas estes sabem onde o alterar.

A resposta correta é: O código torna-se extensível, bastando acrescentar novas funcionalidades, sem ter de alterar código já existente.

Pergunta 16

Incorreto

Pontuou -0,250 de 1,000

 Destacar pergunta

Um "bom" programador adopta um conjunto de boas práticas no que toca ao desenvolvimento de código. Qual das seguintes, não faz parte dessas boas práticas?

Selecione uma opção de resposta:


- ☐ a. Não pretendo responder a esta questão.
- ☐ b. Desenvolver incrementalmente, sem presumir demasiado.
- ☐ c. Registar tempos de desenvolvimento, para melhorar estimativas de esforço.
- ☐ d. Desenvolver o código mais genérico e flexível possível.
- ☒ e. Manter um nível de *technical debt* baixo. ✗

A resposta correta é: Desenvolver o código mais genérico e flexível possível.

Pergunta 17

Correto

Pontuou 1,000 de 1,000

 Destacar pergunta

Que *design patterns* estão presentes no seguinte extrato de código?

```
public class DatabaseConnectionManager {
    private static DatabaseConnectionManager instance;
    private Set<DatabaseConnection> connections;

    private DatabaseConnectionManager() {...}

    public static DatabaseConnectionManager getInstance() {
        if (instance == null)
            instance = new DatabaseConnectionManager();
        return instance;
    }

    /* Returns and removes a connection from the set or
       creates a new one and returns it if set is empty */
    public synchronized DatabaseConnection acquire() {...}

    /* Adds the connection to the set */
    public synchronized void release(DatabaseConnection connection) {...}
}
```

Selecione uma opção de resposta:


- ☐ a. Flyweight e Factory Method
- ☒ b. Singleton e Object Pool ✓
- ☐ c. Factory Method e Singleton
- ☐ d. Factory e Connection Sharing
- ☐ e. Não pretendo responder a esta questão.

A resposta correta é: Singleton e Object Pool

Pergunta 18

Incorreto

Pontuou -0,250 de 1,000

 Destacar pergunta

Em qual destas situações deve ser usado o *design pattern* "Strategy"?

Selecione uma opção de resposta:

- ☐ a. Não pretendo responder a esta questão.
- ☐ b. Quando temos de atacar um problema de várias formas diferentes por questões de eficiência.


- ☐ c. Quando não temos a certeza de qual a melhor estratégia para resolver um problema.
- ☒ d. Quando queremos que um objeto tenha várias interfaces diferentes dependendo do seu contexto. ✖
- ☐ e. Quando queremos que o comportamento dos objetos possa ser alterado em *run time*.

A resposta correta é: Quando queremos que o comportamento dos objetos possa ser alterado em *run time*.

Pergunta 19

Incorreto

Pontuou 0,000 de 1,000

 Destacar pergunta

Qual das seguintes não é uma vantagem do design pattern "Observer"?

Selecione uma opção de resposta:


- ☒ a. Não pretendo responder a esta questão. ✖
- ☐ b. Permite o envio de informação para objectos de vários tipos de uma forma simples.
- ☐ c. Podem ser acrescentados ou removidos observadores a qualquer altura.
- ☐ d. Permite que o objeto que observa tenha acesso ao estado interno do observado.
- ☐ e. Respeita o princípio arquitetural conhecido como "loose coupling".

A resposta correta é: Permite que o objeto que observa tenha acesso ao estado interno do observado.

Pergunta 20

Incorreto

Pontuou -0,250 de 1,000

 Destacar pergunta

Um dos "top" Code Smells é o "Duplicated Method". É possível melhorar o código através de um ou mais *refactorings*, consoante o caso. Quais dos seguintes, não se aplica?

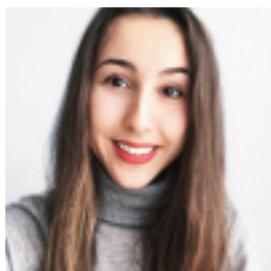
Selecione uma opção de resposta:

- ☐ a. *Pull Up Method*
- ☐ b. *Move Method*
- ☐ c. *Extract Method*
- ☐ d. Não pretendo responder a esta questão.
- ☒ e. *Extract Class* ✖

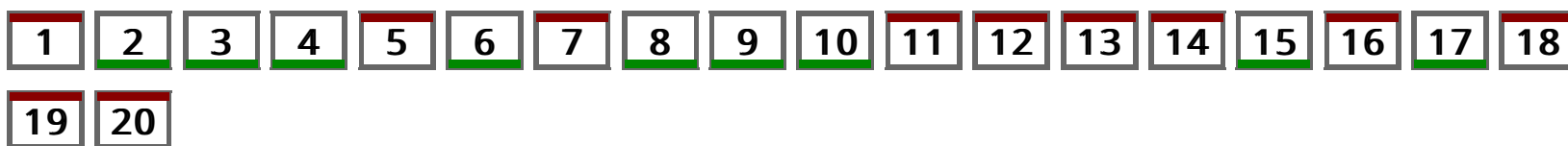
A resposta correta é: *Move Method*

Terminar revisão

NAVEGAÇÃO NO TESTE



Bárbara Sofia Lopez de Carvalho Ferreira da Silva



Terminar revisão

© 2017 UPdigital - Tecnologias Educativas

Nome de utilizador: Bárbara Sofia Lopez de Carvalho Ferreira da Silva (Sair)

Gestão e manutenção da plataforma Moodle U.PORTO da responsabilidade da unidade de Tecnologias Educativas da UPdigital.

Mais informações:

apoio.elearning@uporto.pt | +351 22 040 81 91 | <http://elearning.up.pt>



Based on an original theme created by Shaun Daubney | moodle.org



Minha página principal ► Laboratório de Programação Orientada por Objectos ► Geral ►
2º mini-teste - 9 de junho de 2016 ► Pré-visualização


Pode previsualizar este teste mas se fosse uma tentativa real seria bloqueado porque:


Este teste não está disponível

Pergunta 1

Não respondida

Pontuação 1,000

 Destacar pergunta

 Editar pergunta

Qual dos seguintes não é um tipo primitivo do Java:


Selecione uma opção de resposta:


- ☐ a. Não pretendo responder
- ☐ b. `int`
- ☐ c. `String`
- ☐ d. `char`
- ☐ e. `byte`

Pergunta 2

Não respondida

Pontuação 1,000

 Destacar pergunta

 Editar pergunta

Na linguagem Java, qual dos seguintes métodos não está definido para todos os objetos?


Selecione uma opção de resposta:


- ☐ a. `getClass`
- ☐ b. `compareTo`
- ☐ c. `equals`
- ☐ d. `toString`
- ☐ e. Não pretendo responder

Pergunta 3

Não respondida

Pontuação 1,000

 Destacar pergunta

 Editar pergunta

O que imprime o seguinte extrato de código em Java?

```
public class C1 {  
    public String f() {return "1";}   
    public String g() {return "1";}   
}
```

```
public void print() {System.out.println(f() + g());}

}

public class C2 extends C1 {

    public String f() {return "2"; }

    public String g() {return "2";}

}

C1 obj = new C2();

obj.print();
```


Selecione uma opção de resposta:


- ☐ a. 2
- ☐ b. 11
- ☐ c. Não pretendo responder
- ☐ d. 4
- ☐ e. 22

Pergunta 4

Não respondida

Pontuação 1,000

 Destacar pergunta

 Editar pergunta

Qual das seguintes não é uma palavra chave usada no tratamento de exceções em Java:


Selecione uma opção de resposta:


- ☐ a. **try**
- ☐ b. **fail**
- ☐ c. **throw**
- ☐ d. Não pretendo responder
- ☐ e. **catch**

Pergunta 5

Não respondida

Pontuação 1,000

 Destacar pergunta

 Editar pergunta

Relativamente a boas práticas e ferramentas de teste unitário, selecione a afirmação incorreta:

Selecione uma opção de resposta:


- ☐ a. As ferramentas de teste de mutação como PIT são úteis para avaliar a qualidade dos testes.
- ☐ b. As ferramentas de análise de cobertura de código como EcEmma são fundamentais para conceber testes segundo a abordagem *test-driven development (TDD)*.
- ☐ c. Em JUnit 4 os métodos de teste são anotados com **@Test**.
- ☐ d. Não pretendo responder


- ☐ e. Em JUnit 4, os valores retornados pelos métodos invocados a partir do código de teste são normalmente verificados através do método **assertEquals**.

Pergunta 6

Não respondida

Pontuação 1,000

 Destacar pergunta

 Editar pergunta

Que coleção concreta de Java (classe de implementação) usaria para representar preferências de estudantes por projetos (com a lista ordenada de projetos preferidos por cada estudante)?


Selecione uma opção de resposta:


- ☐ a. **TreeMap<Estudante, HashSet<Projeto>>**
- ☐ b. **LinkedList<Estudante, ArrayList<Projeto>>**
- ☐ c. Não pretendo responder
- ☐ d. **HashMap<Estudante, ArrayList<Projeto>>**
- ☐ e. **HashMap<Estudante, TreeSet<Projeto>>**

Pergunta 7

Não respondida

Pontuação 1,000

 Destacar pergunta

 Editar pergunta

Em Swing, qual dos seguintes é um contentor de nível de topo?


Selecione uma opção de resposta:


- ☐ a. **JPanel**
- ☐ b. **JView**
- ☐ c. Não pretendo responder
- ☐ d. **JList**
- ☐ e. **JFrame**

Pergunta 8

Não respondida

Pontuação 1,000

 Destacar pergunta

 Editar pergunta

Indique a informação incorreta relativamente ao mecanismo de serialização do Java.


Selecione uma opção de resposta:


- ☐ a. As classes serializáveis devem ser anotadas com a anotação **@Serializable**.
- ☐ b. Não pretendo responder
- ☐ c. Quando se solicita a serialização de um objeto com o método **writeObject** da classe **ObjectOutputStream**, esse objeto e todos os por ele referenciados (desde que marcados como serializáveis) são escritos no *stream*.
- ☐ d. Os campos privados também podem ser serializados.
- ☐ e. Os campos que não se pretendem serializar devem ser declarados **transient**.

Pergunta 9

Não respondida

Pontuação 1,000

 Destacar pergunta

 Editar pergunta

Indique a informação incorreta relativamente ao desenvolvimento de aplicações para Android.


Selecione uma opção de resposta:


- ☐ a. Não pretendo responder
- ☐ b. É possível invocar atividades através de objetos do tipo **Intent**.
- ☐ c. A interface com o utilizador para uma atividade é providenciada através de uma hierarquia de vistas — objetos derivados da classe **View**.
- ☐ d. Uma aplicação normalmente compreende várias atividades, subclasses de **Activity**.
- ☐ e. O ponto de entrada numa atividade é indicado pelo seu método **main**.

Pergunta 10

Não respondida

Pontuação 1,000

 Destacar pergunta

 Editar pergunta

Indique a informação incorreta relativamente à utilização de *multithreading* em Java.


Selecione uma opção de resposta:


- ☐ a. A palavra chave **synchronized** pode ser usada para marcar blocos de código sincronizados sobre um dado objeto, evitando assim interferências indesejáveis entre *threads*.
- ☐ b. Em Swing, o estado dos objetos gráficos só deve ser manipulado através do *event dispatching thread*.
- ☐ c. *Multithreading* pode ser usado para desenvolver interfaces gráficas responsivas, capazes de efetuar processamento em *background* e ao mesmo tempo aceitar *input* do utilizador.
- ☐ d. Não pretendo responder
- ☐ e. O número máximo de *threads* numa aplicação Android é igual ao número de *cores* (núcleos) do processador.

Pergunta 11

Não respondida

Pontuação 1,000

 Destacar pergunta

 Editar pergunta

Relativamente ao mecanismo de *profiling* estudado nas aulas, selecione a afirmação correta.


Selecione uma opção de resposta:


- ☐ a. Permite definir *breakpoints* e inspecionar o estado das variáveis do programa quando é alcançado um *breakpoint*.
- ☐ b. Permite guardar e inspecionar mensagens de *debugging*.
- ☐ c. Permite determinar que partes do código consomem mais tempo de execução.
- ☐ d. Não pretendo responder
- ☐ e. Permite detetar código duplicado.

Pergunta 12

Não respondida

Pontuação 1,000

 Destacar pergunta

 Editar pergunta

Indique a afirmação incorreta relativamente ao mecanismo de RMI do Java.

Selecione uma opção de resposta:

- ☐ a. O mecanismo de RMI permite desenvolver aplicações distribuídas em Java, compreendendo normalmente um programa cliente e um programa servidor, em que o primeiro usa serviços disponibilizados pelo segundo.
- ☐ b. O mecanismo de RMI recorre normalmente ao mecanismo de serialização para passar objetos entre programas.
- ☐ c. O mecanismo de RMI permite a um programa invocar métodos sobre objetos remotos (residentes noutro espaço de endereçamento) de forma semelhante à invocação de métodos sobre objetos locais.
- ☐ d. Não pretendo responder
- ☐ e. O mecanismo de RMI suporta comunicação assíncrona entre o programa cliente e o programa servidor por filas de mensagens. Isto é, se o cliente invocar um método do objeto remoto e o servidor estiver em baixo, o pedido é guardado numa fila de espera até o servidor reiniciar.

Pergunta 13

Não respondida

Pontuação 1,000

Destacar pergunta

Editar pergunta

Indique a afirmação incorreta relativamente ao mecanismo de reflexão do Java:

Selecione uma opção de resposta:

- ☐ a. O mecanismo de reflexão permite consultar em tempo de execução meta-informação sobre as classes, métodos, campos e anotações que estão definidos numa aplicação Java.
- ☐ b. Não pretendo responder
- ☐ c. O mecanismo de reflexão permite instanciar classes e invocar métodos que são conhecidos só em tempo de execução (por exemplo, através do nome em string).
- ☐ d. O mecanismo de reflexão permite adicionar métodos a uma classe existente em tempo de execução.
- ☐ e. O mecanismo de reflexão é usado pelo *test runner* do JUnit 4 para identificar e executar os métodos anotados com `@Test`.

Pergunta 14

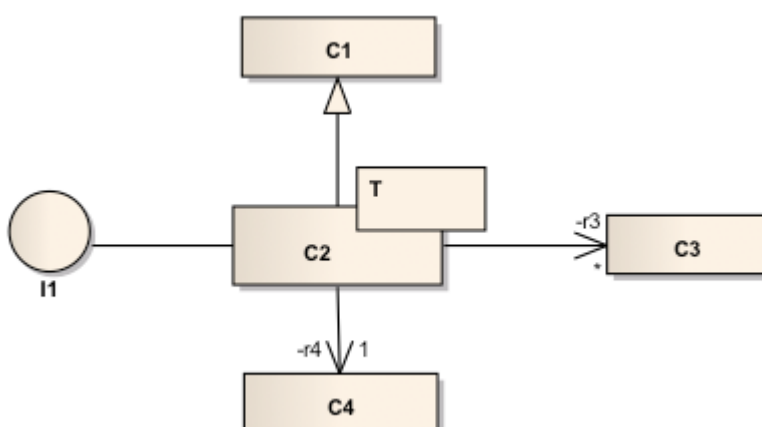
Não respondida

Pontuação 1,000

Destacar pergunta

Editar pergunta

Qual a tradução mais correta em Java da classe C2 do seguinte diagrama de classes UML?




Selecione uma opção de resposta:


- ☐ a. Não pretendo responder
- ☐ b. class C2<T> implements I1 extends C1 { private Set<C3> r3; private C4 r4; }
- ☐ c. class C2<T> implements I1 { private C1 c1; private Set<C3> r3; private C4 r4; }
- ☐ d. class C2<T> implements I1, C1 { private HashSet<C3> r3; private C4 r4; }
- ☐ e. class C2<T> extends I1, C1 { private HashSet<C3> r3; private C4 r4; }

Pergunta 15

Não respondida

Pontuação 1,000

 Destacar pergunta

 Editar pergunta

Que informação pode ser colocada numa transição entre estados num diagrama de estados UML?


Selecione uma opção de resposta:


- ☐ a. ação à entrada (*entry*), evento (*trigger*) e ação à saída (*exit*)
- ☐ b. evento (*trigger*), condição (*guard*) e ação (*effect*)
- ☐ c. pré-condição (*pre-condition*), evento (*trigger*) e pós-condição (*post-condition*)
- ☐ d. evento (*trigger*), condição (*guard*) e atividade (*activity*)
- ☐ e. Não pretendo responder

Pergunta 16

Não respondida

Pontuação 1,000

 Destacar pergunta

 Editar pergunta

Que elementos principais podem fazer parte de diagramas de sequência UML?


Selecione uma opção de resposta:


- ☐ a. Linhas de vida, mensagens e fragmentos combinados.
- ☐ b. Não pretendo responder
- ☐ c. Linhas de vida, transições e fragmentos combinados.
- ☐ d. Atores, mensagens e operadores de interação.
- ☐ e. Linhas de vida, transições e operadores de interação.

Pergunta 17

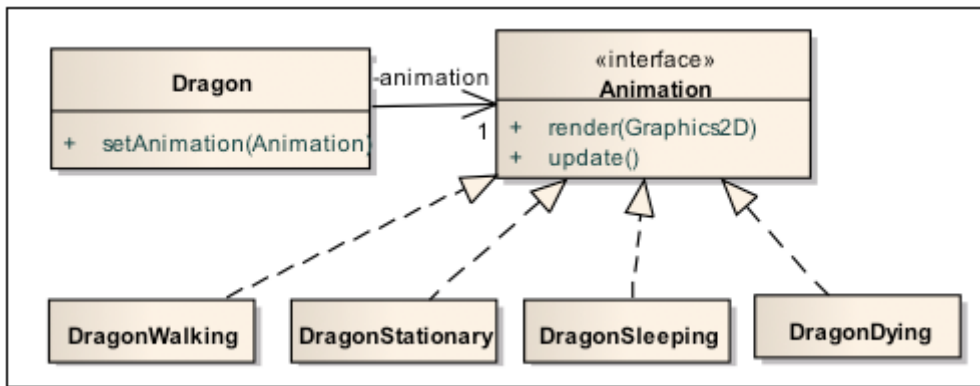
Não respondida

Pontuação 1,000

 Destacar pergunta

 Editar pergunta

O diagrama seguinte reflete que padrão de desenho?



Selecione uma opção de resposta:

- ☐ a. VISITOR
- ☐ b. STRATEGY
- ☐ c. STATE
- ☐ d. Não pretendo responder
- ☐ e. SINGLETON

Pergunta 18

Não respondida

Pontuação 1,000

Destacar pergunta

Editar pergunta

Que padrão de desenho está a ser aplicado no seguinte código?

```
public abstract class C {
    public abstract void f();
    public abstract void g();
    public void h() {f(); g();}
}
```

Selecione uma opção de resposta:

- ☐ a. COMPOSITE
- ☐ b. TEMPLATE METHOD
- ☐ c. VISITOR
- ☐ d. STRATEGY
- ☐ e. Não pretendo responder

Pergunta 19

Não respondida

Pontuação 1,000

Destacar pergunta

Editar pergunta

Que *code smell* não está presente no seguinte extrato de código?

```

public class Game {
    ...
    public boolean moveHero(Direction move){
        if(move == Direction.UP){ // move up
            int newPosX = hero.getX();
            int newPosY = hero.getY() - 1;
            hero.setCoord(newPosX, newPosY);
            return true;
        }
        else if(move == Direction.LEFT) { // move left
            int newPosX = hero.getX() - 1;
            int newPosY = hero.getY();
            hero.setCoord(newPosX, newPosY);
            return true;
        }
    }
}

```


Selecione uma opção de resposta:


- ☐ a. Não pretendo responder
- ☐ b. *Feature Envy*
- ☐ c. *Temporary Field*
- ☐ d. *Duplicated Code*
- ☐ e. *Comments*

Pergunta 20

Não respondida

Pontuação 1,000

 Destacar pergunta

 Editar pergunta

Que *refactoring* pode ser aplicado para melhorar o seguinte código?

```

public class Game {
    ...
    public boolean equalPositions(Position p1,
                                   Position p2){
        return p1.getY() == p2.getY()
            && p1.getX() == p2.getX();
    }
    ...
}

```

Selecione uma opção de resposta:

- ☐ a. *Move Method* (para classe **Position**)
- ☐ b. *Pull Up Method* (para classe **Position**)
- ☐ c. Não pretendo responder
- ☐ d. *Extract Method*

☐ e. Introduce Null Object

Seguinte

ADMINISTRAÇÃO DO TESTE

- Editar configurações
- Revogações por grupo
- Revogações por utilizador

⚙ Gerir perguntas do teste

🔍 Pré-visualização

Resultados

- Permissões
- Verificar permissões
- Filtros
- Registos de atividade
- Cópia de segurança
- Restaurar

Base de dados de perguntas

ADMINISTRAÇÃO DA PÁGINA/UNIDADE

ASSUMIR O CARGO DE...

O MEU PERFIL

NAVEGAÇÃO NO TESTE



Nuno Honório Rodrigues Flores

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20


Terminar resolução...

Tempo restante **0:31:46**

Iniciar nova previsualização

© 2017 U.Porto - Tecnologias Educativas

Nome de utilizador: Nuno Honório Rodrigues Flores (Sair)

 Documentação Moodle para esta página

Gestão e manutenção da plataforma Moodle U.PORTO da responsabilidade da unidade de Tecnologias Educativas. Mais informações:
apoio.elearning@uporto.pt | +351 22 040 81 91 | <http://elearning.up.pt>



Based on an original theme created by Shaun Daubney | moodle.org

LABORATÓRIO DE PROGRAMAÇÃO ORIENTADA POR OBJECTOS

Data de início	Sexta, 12 Junho 2015, 09:01
Estado	Terminada
Completo em	Sexta, 12 Junho 2015, 09:42
Tempo gasto	41 minutos 23 segundos

Pergunta 1

Correto

Pontuação 1,00

Destacar pergunta

Relativamente aos princípios fundamentais de orientação por objetos, selecione a afirmação incorreta:

Selecione uma opção de resposta:

☐

a. Polimorfismo refere-se à possibilidade da implementação de um método variar de acordo com a subclasse do objeto.

☒

b. Encapsulamento refere-se à possibilidade de definir classes públicas dentro doutras classes. ✓

☐

c. Abstração refere-se à possibilidade de definir classes como abstrações para conjuntos de objetos com propriedades similares.

☐

d. Herança refere-se à possibilidade das subclasses herdarem propriedades das superclasses.

☐

e. Não quero responder a esta questão.

A resposta correta é:Encapsulamento refere-se à possibilidade de definir classes públicas dentro doutras classes.

Pergunta 2

Correto

Pontuação 1,00

Destacar pergunta

Qual das seguintes *keywords* não é usada no tratamento de exceções em Java:

Selecione uma opção de resposta:

☐

a. catch

☒

b. onerror ✓

☐

c. Não quero responder a esta questão.

☐

d. try

☐

e. throw

A resposta correta é:onerror

Pergunta 3

Incorreto

Pontuação 1,00

Destacar pergunta

Relativamente a boas práticas e ferramentas de teste unitário, selecione a afirmação incorreta:

Selecione uma opção de resposta:

☒

a. Na abordagem *test-driven development*, os testes devem ser escritos antes do código a testar. ✗

☐

b. As ferramentas de análise de cobertura de código como EcEmma são usadas para identificar partes do código que não são exercitadas pelos testes.

☐

c. Não quero responder a esta questão.

☐

d. Nos *frameworks* de teste unitário como JUnit, os valores retornados pelos métodos invocados a partir do código de teste são normalmente verificados através de asserções.

☐

e. As ferramentas de teste de mutação como PIT são usadas para gerar automaticamente novos testes, por mutação de testes existentes com base em algoritmos genéticos.

A resposta correta é:As ferramentas de teste de mutação como PIT são usadas para gerar automaticamente novos testes, por mutação de testes existentes com base em algoritmos genéticos.

Pergunta 4

Incorreto

Pontuação 1,00

Destacar pergunta

Qual das seguintes funcionalidades não é suportada por Java 8?

Selecione uma opção de resposta:

☐

a. Não quero responder a esta questão.

☒

b. Expressões lambda. ✗

☐

c. Herança múltipla, isto é, a possibilidade de uma classe estender várias classes.

☐

d. Referências para métodos.

☐

e. Tipos e métodos genéricos (ou parametrizados).

A resposta correta é:Herança múltipla, isto é, a possibilidade de uma classe estender várias classes.

Pergunta 5

Incorreto

Pontuação 1,00

Destacar pergunta

Que coleção concreta de Java (classe de implementação) usaria para representar uma coleção ordenada de números em vírgula flutuante de precisão dupla, sem duplicados?

Selecione uma opção de resposta:

- ☒ a. HashSet<double> ❌
- ☐ b. TreeMap<Double>
- ☐ c. TreeSet<Double>
- ☐ d. ArrayList<double>
- ☐ e. Não quero responder a esta questão.

A resposta correta é:TreeSet<Double>

Pergunta 6

Correto

Pontuação 1,00

Destacar pergunta

Relativamente aos listeners de Swing, selecione a afirmação incorreta:

Selecione uma opção de resposta:

- ☐ a. Permitem programar comportamentos em resposta a eventos do utilizador.
- ☐ b. Não quero responder a esta questão.
- ☐ c. Seguem o padrão Observer.
- ☐ d. Permitem implementar a parte “C” do padrão MVC.
- ☒ e. São usados para implementar interfaces com reconhecimento de voz. ✔️

A resposta correta é:São usados para implementar interfaces com reconhecimento de voz.

Pergunta 7

Correto

Pontuação 1,00

Destacar pergunta

Qual a finalidade do mecanismo de serialização do Java?

Selecione uma opção de resposta:

- ☐ a. Permitir aceder a ficheiros de forma sequencial.
- ☐ b. Permitir executar de forma segura blocos de código concorrentes.
- ☐ c. Permitir controlar as versões das classes.
- ☒ d. Converter facilmente um objeto (e todos os por ele referenciados) para uma sequência de bytes que pode ser enviada para um stream (ficheiro, rede, etc.). ✔️
- ☐ e. Não quero responder a esta questão.

A resposta correta é:Converter facilmente um objeto (e todos os por ele referenciados) para uma sequência de bytes que pode ser enviada para um stream (ficheiro, rede, etc.).

Pergunta 8

Correto

Pontuação 1,00

Destacar pergunta

Qual das seguintes não é uma aplicação típica de multithreading em Java?

Selecione uma opção de resposta:

- ☐ a. Desenvolver interfaces gráficas responsivas, capazes de efetuar processamento em background e ao mesmo tempo aceitar input do utilizador.
- ☒ b. Desenvolver aplicações em Android que se adaptam facilmente às dimensões de múltiplos dispositivos. ✔️
- ☐ c. Tirar partido de processadores multi-core.
- ☐ d. Realização de garbage collection pela máquina virtual do Java, em paralelo com a execução normal do programa.
- ☐ e. Não quero responder a esta questão.

A resposta correta é:Desenvolver aplicações em Android que se adaptam facilmente às dimensões de múltiplos dispositivos.

Pergunta 9

Correto

Pontuação 1,00

Destacar pergunta

Relativamente ao mecanismo de RMI do Java, selecione a afirmação incorreta:

Selecione uma opção de resposta:

- ☐ a. O mecanismo de RMI permite a um programa invocar métodos sobre objetos remotos (residentes num espaço de endereçamento diferente) de forma semelhante à invocação de métodos sobre objetos locais.
- ☒ b. O mecanismo RMI não suporta callbacks, isto é, chamadas do servidor para o cliente. ✔️
- ☐ c. O mecanismo de RMI recorre normalmente ao mecanismo de serialização para passar objetos entre programas.
- ☐ d. Não quero responder a esta questão.
- ☐ e. O mecanismo de RMI permite desenvolver aplicações distribuídas em Java, compreendendo normalmente um programa cliente e um programa servidor, em que o primeiro usa serviços disponibilizados pelo segundo.

A resposta correta é: O mecanismo RMI não suporta *callbacks*, isto é, chamadas do servidor para o cliente.

Pergunta 10

Correto

Pontuação 1,00

Destacar
pergunta

Relativamente ao mecanismo de *profiling* estudado nas aulas, selecione a afirmação correta:

Selecione uma opção de resposta:

- ☐ a. Permite determinar que partes do código são cobertas pelos testes.
- ☐ b. Não quero responder a esta questão.
- ☒ c. Permite determinar que métodos consomem mais tempo de execução. ✓
- ☐ d. Permite definir perfis de acesso a uma aplicação.
- ☐ e. Permite detetar *code smells* no código.

A resposta correta é: Permite determinar que métodos consomem mais tempo de execução.

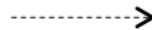
Pergunta 11

Correto

Pontuação 1,00

Destacar
pergunta

Nos diagramas de classes UML, o seguinte símbolo representa que tipo de relação entre classes?



Selecione uma opção de resposta:

- ☐ a. associação navegável
- ☒ b. dependência ✓
- ☐ c. Não quero responder a esta questão.
- ☐ d. generalização.
- ☐ e. concretização (*realization*)

A resposta correta é: dependência

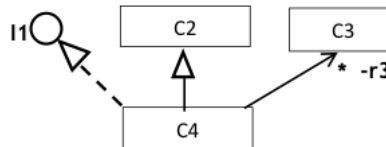
Pergunta 12

Correto

Pontuação 1,00

Destacar
pergunta

Qual a tradução mais correta em Java da classe C4 do seguinte diagrama de classes UML?



Selecione uma opção de resposta:

- ☐ a. class C4 extends I1, C2 { private HashSet<C3> r3; }
- ☐ b. class C4<I1> extends C2 { private Set<C3> r3; }
- ☐ c. Não quero responder a esta questão.
- ☒ d. class C4 implements I1 extends C2 { private HashSet<C3> r3; } ✓
- ☐ e. class C4 implements I1 { private C2 c2; private Set<C3> r3; }

A resposta correta é: class C4 implements I1 extends C2 { private HashSet<C3> r3; }

Pergunta 13

Correto

Pontuação 1,00

Destacar
pergunta

Qual dos seguintes não é um elemento utilizável em diagramas de sequência UML?

Selecione uma opção de resposta:

- ☐ a. linha de vida (*lifeline*).
- ☒ b. transição. ✓
- ☐ c. fragmento combinado.
- ☐ d. mensagem.
- ☐ e. Não quero responder a esta questão.

A resposta correta é: transição.

Pergunta 14

Incorreto

Relativamente aos diagramas de estados UML, selecione a afirmação incorreta:

Pontuação 1,00

Destacar pergunta

Selecione uma opção de resposta:

- ☒ a. Não quero responder a esta questão. ✖
- ☐ b. A utilização de regiões ortogonais permite evitar a explosão combinatória de estados.
- ☐ c. Podem-se utilizar estados compostos, contendo subestados sequenciais.
- ☐ d. Não podem existir múltiplas transições com o mesmo evento e o mesmo estado de origem.
- ☐ e. As transições são instantâneas.

A resposta correta é:Não podem existir múltiplas transições com o mesmo evento e o mesmo estado de origem.

Pergunta 15

Correto

Pontuação 1,00

Destacar pergunta

“Defines a family of algorithms, encapsulates each one, and make them interchangeable. It lets the algorithm vary independently from clients that use it.” Qual o padrão de desenho que tem este propósito (intent)?

Selecione uma opção de resposta:

- ☐ a. BUILDER
- ☒ b. STRATEGY ✔
- ☐ c. Não quero responder a esta questão.
- ☐ d. TEMPLATE METHOD
- ☐ e. COMPOSITE

A resposta correta é:STRATEGY

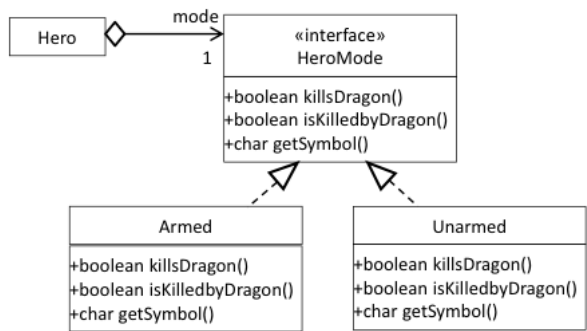
Pergunta 16

Correto

Pontuação 1,00

Destacar pergunta

O diagrama abaixo reflete que padrão de desenho?



Selecione uma opção de resposta:

- ☒ a. STATE ✔
- ☐ b. Não quero responder a esta questão.
- ☐ c. TEMPLATE METHOD
- ☐ d. MODAL
- ☐ e. VISITOR

A resposta correta é:STATE

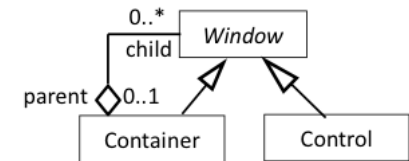
Pergunta 17

Correto

Pontuação 1,00

Destacar pergunta

O diagrama abaixo reflete que padrão de desenho?



Selecione uma opção de resposta:

- ☐ a. BUILDER
- ☐ b. CONTROLLER
- ☐ c. Não quero responder a esta questão.
- ☒ d. COMPOSITE ✔
- ☐ e. HIERARCHY

A resposta correta é:COMPOSITE

Pergunta 18

Correto

Pontuação 1,00

Destacar
pergunta

Em que consiste *refactoring*?

Selecione uma opção de resposta:

- ☐ a. Efetuar alterações a um programa para adicionar novas funcionalidades.
- ☒ b. Efetuar alterações à estrutura interna de um programa para o tornar mais fácil de compreender e modificar, sem alterar o comportamento observável do programa. ✓
- ☐ c. Alterar o código de um programa para ficar coerente com um diagrama de classes.
- ☐ d. Efetuar alterações a um programa para corrigir *bugs* e garantir que passa nos testes.
- ☐ e. Não quero responder a esta questão.

A resposta correta é:Efetuar alterações à estrutura interna de um programa para o tornar mais fácil de compreender e modificar, sem alterar o comportamento observável do programa.

Pergunta 19

Correto

Pontuação 1,00

Destacar
pergunta

Qual dos seguintes *code smells* não pode ser resolvido pelo *refactoring Extract Method*?

Selecione uma opção de resposta:

- ☐ a. Switch Statements
- ☐ b. Long Method
- ☐ c. Duplicated Code
- ☒ d. Lazy Class ✓
- ☐ e. Não quero responder a esta questão.

A resposta correta é:Lazy Class

Pergunta 20

Incorreto

Pontuação 1,00

Destacar
pergunta

Qual dos seguintes *code smells* não está presente no seguinte código?

```
public class Pessoa {
    private String n1; // primeiro nome
    private String n2; // ultimo nome
    private String aux;
    private int i; // idade

    public String toString() {
        aux = n1 + " " + n2;
        return "nome completo: " + aux + " idade: " + i;
    }

    public String getFullName() {
        aux = n1 + " " + n2;
        return aux;
    } ...
}
```

Selecione uma opção de resposta:

- ☐ a. Não quero responder a esta questão.
- ☐ b. *Duplicated code*
- ☒ c. *Temporary Field* ✗
- ☐ d. *Data Clumps*
- ☐ e. *Comments*

A resposta correta é:*Data Clumps*



Gestão e manutenção da plataforma Moodle U.PORTO da responsabilidade da TE - Unidade de Tecnologias Educativas

Tema gráfico criado por idd.fba.up.pt

Nome de utilizador: [Pedro Miguel Vieira da Silva](#). ([Sair](#))

Laboratório de Programação Orientada por Objetos (MIEIC)

7 de Junho de 2013

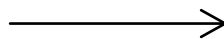
Mini-teste de escolha múltipla - Sem consulta - 60 Minutos

Assinala com um "X" a resposta correta a cada uma das perguntas seguintes. Cada pergunta só tem uma resposta correta. Respostas certas valem 1 ponto. Respostas erradas descontam 0.25 pontos. Para apagar uma resposta anteriormente introduzida, selecione a opção "Não quero responder a esta questão".

Nome: _____

Diagramas de Classes UML

1. Nos diagramas de classes UML, o seguinte símbolo representa que tipo de relação entre classes?

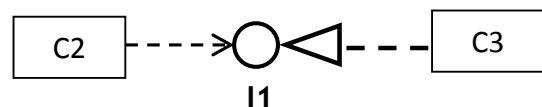


<input type="checkbox"/>	a. associação navegável
<input type="checkbox"/>	b. concretização (<i>realization</i>)
<input type="checkbox"/>	c. generalização
<input type="checkbox"/>	d. dependência

2. Nos diagramas de classes UML, a relação de generalização permite relacionar que tipos de elementos?

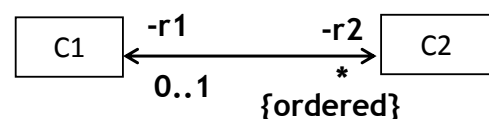
<input type="checkbox"/>	a. subclasses e superclasses
<input type="checkbox"/>	b. classes e interfaces
<input type="checkbox"/>	c. objetos e classes
<input type="checkbox"/>	d. instâncias de classes genéricas e classes genéricas

3. Qual é o significado do seguinte diagrama de classes UML:



<input type="checkbox"/>	a. a classe C2 usa uma interface I1 que é implementada pela classe C3
<input type="checkbox"/>	b. a classe C2 implementa a interface I1 e a classe C3 tem uma associação com I1
<input type="checkbox"/>	c. a classe C2 depende da interface I1 e a classe C3 tem uma associação com I1
<input type="checkbox"/>	d. a classe C2 interage com a classe C3 pela porta I1

4. Qual a tradução mais correta em Java do seguinte diagrama de classes UML:



<input type="checkbox"/>	a. class C1 { private List<C2> r2; }	class C2 { private C1 r1; }
<input type="checkbox"/>	b. class C1 { private Set<C2> r2; }	class C2 { private C1 r1; }
<input type="checkbox"/>	c. class C1 { private C2 r2; }	class C2 { private List<C1> r1; }
<input type="checkbox"/>	d. class C1 { private Queue<C2> r2; }	class C2 { private C1 r1; }

Diagramas de Sequência UML

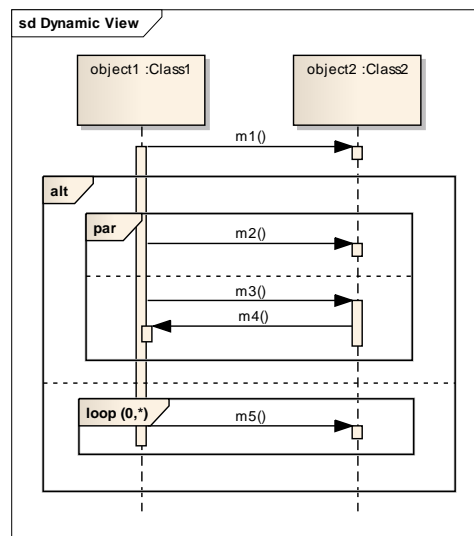
5. Os diagramas de sequência UML são mais apropriados para:

	a. mostrar sequências de instruções
	b. mostrar os estados por que os objetos podem passar ao longo da sua vida
	c. mostrar padrões de trocas de mensagens entre objetos num determinado contexto
	d. mostrar uma configuração de objetos e ligações entre objetos

6. Qual dos seguintes não é um tipo de mensagem válida em diagramas de sequência UML?

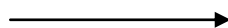
	a. mensagem de chamada de operação
	b. Mensagem de retorno de operação
	c. Mensagem de envio de sinal
	d. Mensagem de criação de classe

7. Qual das seguintes sequências de mensagens não é permitida pelo seguinte diagrama de sequência?



	a. m1
	b. m1 m3 m2 m4
	c. m1 m5 m5 m5
	d. m1 m4 m3 m2

8. Nos diagramas de sequência UML, o seguinte símbolo representa que tipo de mensagem?



	a. mensagem síncrona
	b. mensagem assíncrona
	c. mensagem de criação de objeto
	d. mensagem de fluxo de dados

Diagramas de Estados UML

9. Os diagramas de estados UML são mais apropriados para

	a. modelar o ciclo de vida de objetos ou sistemas
	b. modelar a interação entre objetos
	c. modelar algoritmos
	d. enumerar as variáveis de estado de um objeto

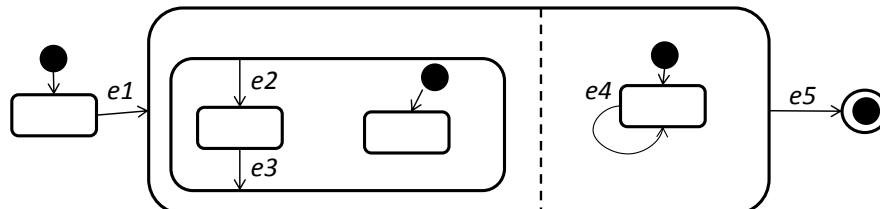
10. Nos diagramas de estados UML, a utilização de regiões ortogonais é útil para:

	a. evitar a explosão combinatória de estados
	b. evitar a explosão combinatória de transições
	c. evitar a explosão combinatória de eventos
	d. modelar eventos temporais absolutos e relativos

11. Nos diagramas de estados UML, qual dos seguintes elementos não pode ser usado numa transição?

	a. ação
	b. condição de guarda
	c. atividade
	d. evento temporal

12. Quais das seguintes sequências de eventos não é aceite pelo seguinte diagrama de estados?



	a. e1 e4 e3 e5
	b. e1 e2 e5
	c. e1 e2 e3 e4 e5
	d. e1 e4 e2 e2 e4 e3 e5

Padrões de Desenho

13. Um padrão de desenho surge com o propósito de:

	a. Reutilizar software para poupar tempo de desenvolvimento.
	b. Resolver problemas recorrentes de uma forma universal.
	c. Sugerir soluções garantidamente boas para problemas recorrentes.
	d. Ajudar no refactoring de código legado.

14. Os padrões de desenho assentam nas “pedras basilares” da orientação por objetos. Estas são:

	a. Abstração, Polimorfismo, Herança e Encapsulamento
	b. Abstração, Herança, Reencaminhamento e Polimorfismo
	c. Abstração, Polimorfismo, Invocação Remota e Reencaminhamento
	d. Abstração, Herança, Invocação Remota e Polimorfismo

15. A melhor estratégia para usar padrões de desenho é:

	a. Usá-los sempre que possível, pois serão sempre uma vantagem.
	b. Tê-los presentes para os usar sempre que a necessidade surja.
	c. Usar só se for preciso fazer refactoring.
	d. Usar apenas se já tivermos tentado tudo para resolver o problema em mãos.

16. Quando usamos o SWING em Java para construir interfaces gráficas, quais os dois padrões de desenho mais utilizados?

	a. COMPOSITE e OBSERVER
	b. STRATEGY e VISITOR
	c. VISITOR e COMPOSITE
	d. OBSERVER e STRATEGY

Refactoring e Code Smells

17. Em que consiste *refactoring*?

	a. Efetuar alterações à estrutura interna de um programa para o tornar mais fácil de compreender e modificar, sem alterar o comportamento observável do programa.
	b. Efetuar alterações à estrutura interna de um programa para melhorar o comportamento observável do programa.
	c. Efetuar alterações à estrutura interna de um programa para garantir que passa nos testes.
	d. Efetuar alterações à formatação de um programa para o tornar mais fácil de compreender e modificar, sem alterar o comportamento observável do programa.

18. Qual dos seguintes *code smells* não pode ser resolvido pelo *refactoring Extract Method*?

	a. Long Method
	b. Switch Statements
	c. Duplicated Code
	d. Lazy Class

19. Que *refactoring* considera ser o mais útil aplicar para melhorar o seguinte código?

```
if (hX==dX) {
    if (Math.abs(dY-hY) == 1) {
        if (h.armado) {
            d.vivo= false;
            labi[dX][dY]= ' ';
        }
        else if (d.dorme == false)
            h.vivo=false;
    }
}
....
if (hY==dY) {
    if (Math.abs(dX-hX) == 1) {
        if (h.armado) {
            d.vivo= false;
            labi[dX][dY]= ' ';
        }
        else if (d.dorme == false)
            h.vivo=false;
    }
}
```

<input type="radio"/>	a. Extract Method
<input type="radio"/>	b. Pull-Up Method
<input type="radio"/>	c. Replace Conditional with Polymorphism
<input type="radio"/>	d. Undo Copy/Paste

20. Que *code smell* existe no seguinte código?

```
public class Customer {
    private Phone mobilePhone;

    public String getMobilePhoneNumber() {
        return "(" + mobilePhone.getAreaCode() + ") " +
            mobilePhone.getPrefix() + "-" +
            mobilePhone.getNumber();
    } ...
}
```

<input type="radio"/>	a. <i>Feature Envy</i>
<input type="radio"/>	b. <i>Lazy Class</i>
<input type="radio"/>	c. <i>Refused Bequest</i>
<input type="radio"/>	d. <i>Duplicated Code</i>

Laboratório de Programação Orientada por Objetos (MIEIC)

8/6/2012

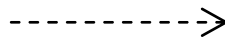
Mini-teste de escolha múltipla - Sem consulta - 60 Minutos

Assinala com um "X" a resposta correta a cada uma das perguntas seguintes. Cada pergunta só tem uma resposta correta. Respostas certas valem 1 ponto. Respostas erradas descontam 0.25 pontos.

Nome: _____

Diagramas de Classes UML

1. Nos diagramas de classes UML, o seguinte símbolo representa que tipo de relação entre classes?



<input type="checkbox"/>	a. associação navegável
<input type="checkbox"/>	b. concretização (<i>realization</i>)
<input type="checkbox"/>	c. generalização
<input type="checkbox"/>	d. dependência

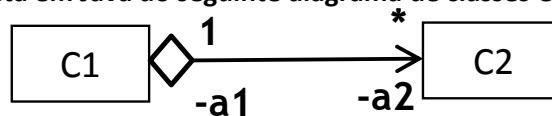
2. Nos diagramas de classes UML, a relação de concretização (*realization*) permite relacionar que tipos de elementos?

<input type="checkbox"/>	a. objetos e classes
<input type="checkbox"/>	b. classes concretas e classes abstratas
<input type="checkbox"/>	c. classes e interfaces
<input type="checkbox"/>	d. classes não genéricas e classes genéricas

3. Nos diagramas de classes UML, membros estáticos (atributos e operações) são representados:

<input type="checkbox"/>	a. em itálico
<input type="checkbox"/>	b. com sublinhado
<input type="checkbox"/>	c. com um estereótipo
<input type="checkbox"/>	d. com "/" antes do nome

4. Qual a tradução mais correta em Java do seguinte diagrama de classes UML:



<input type="checkbox"/>	a. <code>class C1 { private Set<C2> a2; } class C2 { }</code>
<input type="checkbox"/>	b. <code>class C1 { private C2 a2; } class C2 { private HashSet<C1> a1; }</code>
<input type="checkbox"/>	c. <code>class C1 { private List<C2> a2; } class C2 { private C1 a1; }</code>
<input type="checkbox"/>	d. <code>class C1 { private List<C2> a2; class C2 { }}</code>

Diagramas de Sequência UML

5. Os diagramas de sequência UML são mais apropriados para:

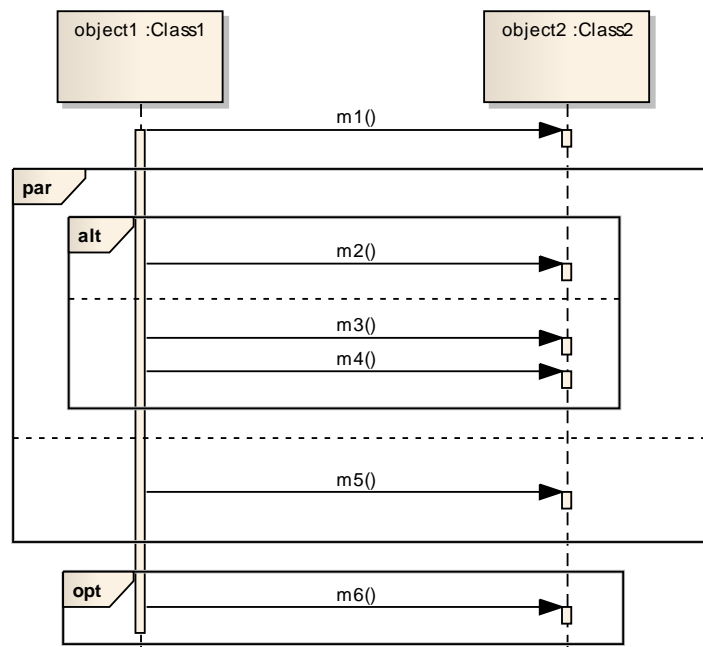
	a. Mostrar os passos envolvidos num algoritmo.
	b. Mostrar o ciclo de vida de objetos.
	c. Mostrar interações entre objetos num determinado contexto.
	d. Mostrar uma configuração de objetos e ligações entre objetos.

6. Que tipos de elementos podem ter linhas de vida num diagrama de sequência?

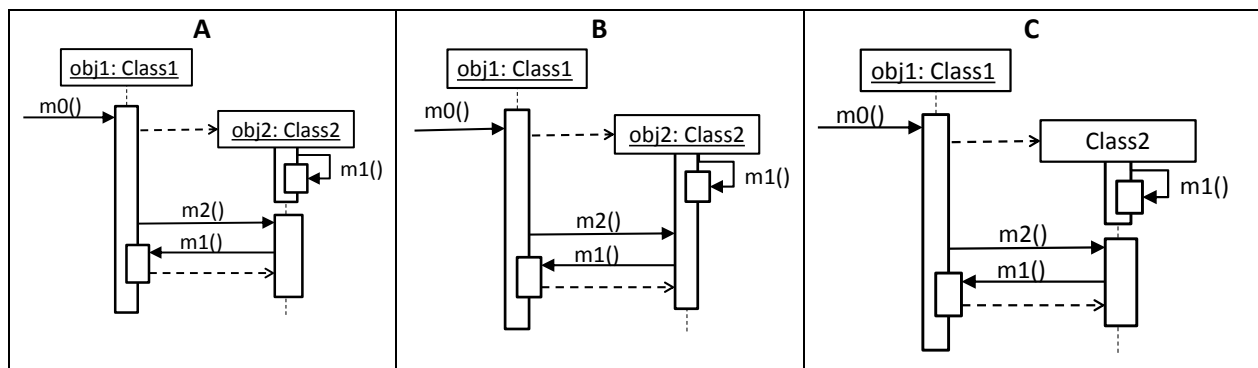
	a. Atores, classes, instâncias de atores, instâncias de classes e casos de utilização.
	b. Atores, classes, instâncias de atores e instâncias de classes.
	c. Qualquer elemento UML.
	d.. Apenas classes e instâncias de classes.

7. Qual das sequências de mensagens não é permitida pelo seguinte diagrama de sequência UML?

	a. m1 m5 m2
	b. m1 m3 m5 m4
	c. m1 m3 m5 m6
	d. m1 m2 m5 m6



8. Quais dos seguintes diagramas de sequência UML são sintaticamente válidos? (Nota: A mensagem *m0()* é válida)



	a. Nenhum
	b. Todos
	c. Só A
	d. Só A e C

Diagramas de Estados UML

9. Nos diagramas de estados UML, as transições podem ser etiquetadas com que elementos?

	a. evento, condição de guarda e ação
	b. evento, condição de guarda e atividade
	c. nome, evento, pré-condição e pós-condição
	d. evento e condição inicial

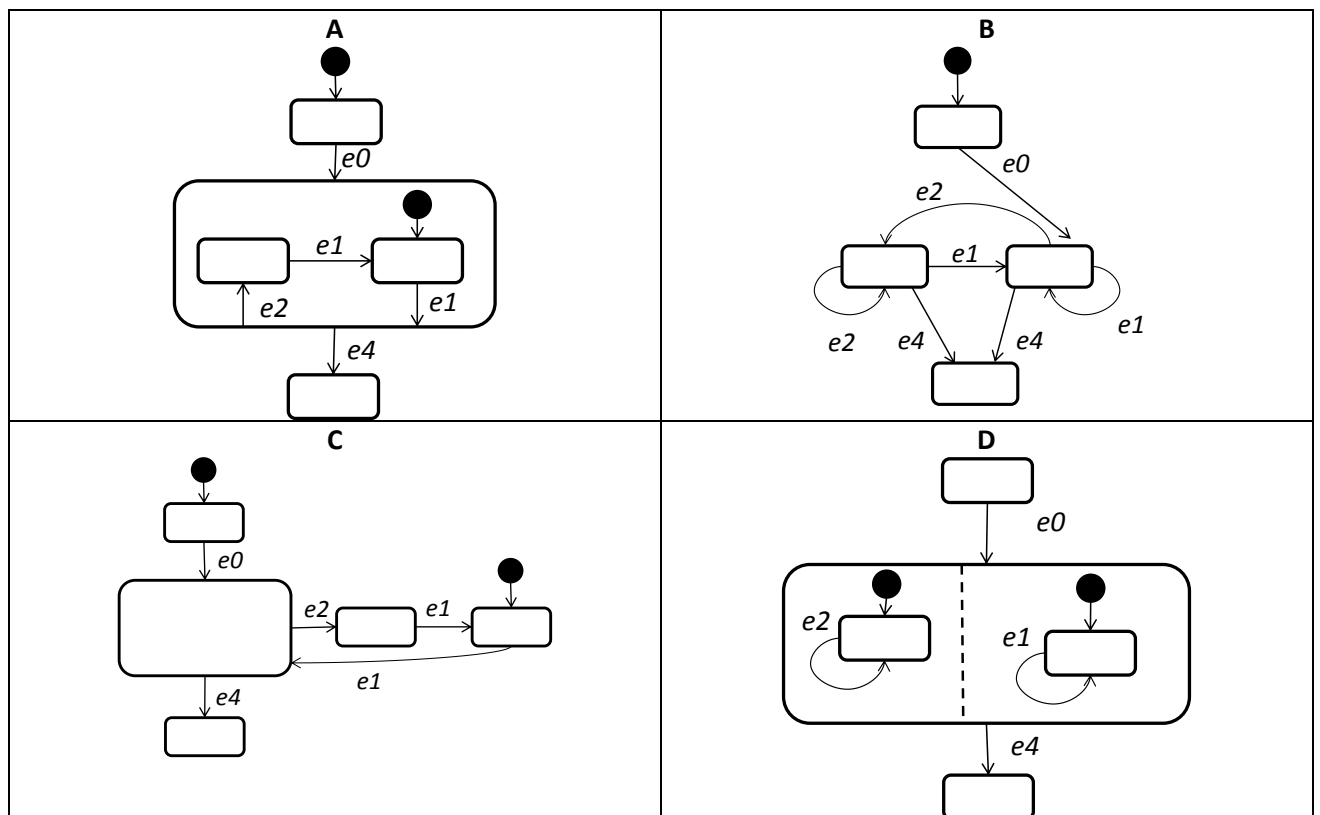
10. Nos diagramas de estados UML, estados compostos (com subestados sequenciais) são úteis para:

	a. evitar a explosão combinatório de estados
	b. evitar a explosão combinatório de transições
	c. modelar comportamentos repetitivos
	d. modelar eventos temporais

11. Nos diagramas de estados UML, quais dos seguintes tipos de eventos são aceites?

	a. chamada de operações, entrada em estado, saída de estado, início e fim
	b. chamada de operações, retorno de operações, eventos temporais e sinais
	c. chamada de operações, exceções, e <i>listeners</i>
	d. chamada de operações e eventos temporais

12. Quais dos seguintes diagramas de estados UML são equivalentes, no sentido de aceitarem as mesmas sequências de eventos?



	a. A e D
	b. A e C
	c. A, B e D
	d. A, B e C

Padrões de Desenho

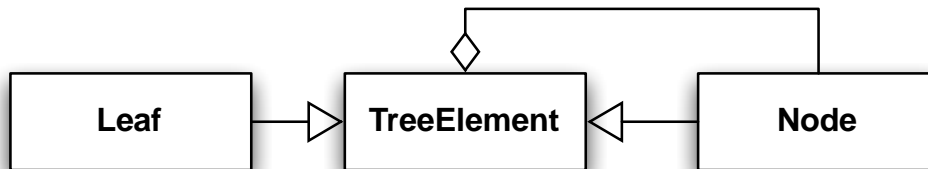
13. “Defines a family of algorithms, encapsulates each one, and make them interchangeable. It lets the algorithm vary independently from clients that use it.” Qual o padrão de desenho que tem este propósito (intent)?

	a. COMPOSITE
	b. STRATEGY
	c. TEMPLATE METHOD
	d. BUILDER

14. “Defines an interface for creating objects, but let subclasses to decide which class to instantiate.” Qual o padrão de desenho que tem este propósito (intent)?

	a. BUILDER
	b. ITERATOR
	c. OBSERVER
	d. FACTORY METHOD

15. O diagrama abaixo reflecte que padrão de desenho?



	a. BUILDER
	b. TEMPLATE METHOD
	c. COMPOSITE
	d. FACTORY METHOD

16. Durante o desenvolvimento do primeiro projecto guiado, para se proceder ao desenho do labirinto e respectivos elementos gráficos, criava-se uma subclasse de JPanel e fazia-se *override* ao método *paintComponent()*. Qual o padrão usado pelo SWING que permite este tipo de mecanismo?

	a. COMPOSITE
	b. TEMPLATE METHOD
	c. FACTORY METHOD
	d. OBSERVER

Refactoring e Code Smells

17. Quando não se deve fazer *refactoring* a uma parte específica de código?

	a. quando se está a adicionar uma nova funcionalidade
	b. quando se está a resolver um bug ou anomalia
	c. quando se está a analisar esse código e a tentar perceber como funciona
	d. quando se está a meio duma implementação e o código ainda não funciona

18. Que *refactoring* considera ser o mais útil aplicar para melhorar o seguinte código?

```
double getSpeed() {
    switch (_type) {
        case EUROPEAN:
            return getBaseSpeed();
        case AFRICAN:
            return getBaseSpeed() - getLoadFactor() * _numberOfCoconuts;
        case NORWEGIAN_BLUE:
            return (_isNailed) ? 0 : getBaseSpeed(_voltage);
    } throw new RuntimeException ("Should be unreachable");
}
```

	a. Extract Method
	b. Switch Statements
	c. Replace Conditional with Polymorphism
	d. Duplicated code

19. Que *code smells* podem ser resolvidos pelo *refactoring Extract Method*?

	a. Long Method e Switch Statements
	b. Duplicated Code e Comments
	c. Long Parameter List e Comments
	d. Large Class e Data Clumps

20. Que *refactoring* sugere utilizar para melhorar o seguinte código?

```
public int _type;
```

	a. Pull Up Field
	b. Extract Method
	c. Encapsulate Field
	d. Public Field