

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.

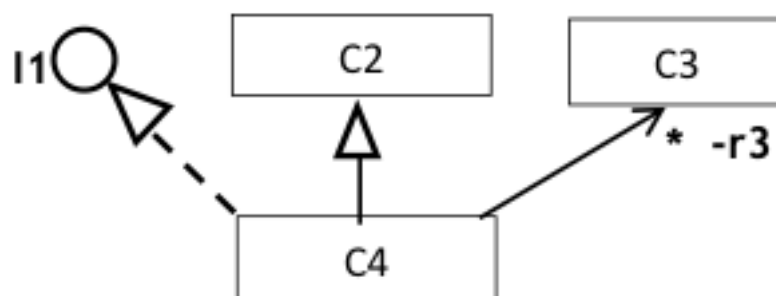
Pergunta 8

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 arquitectural 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?

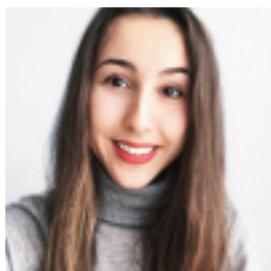
Selecione uma opção de resposta:

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

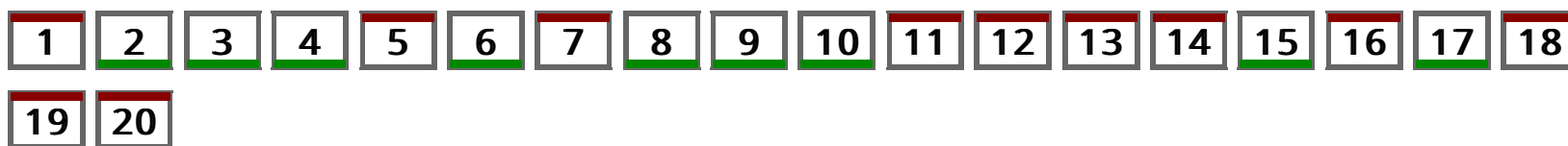
A resposta correta é: *Move Method*

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