CADERNO DE QUESTÕES

HOSPITAL DE CLÍNICAS
PORTO ALEGRE RS

1971 • 2011 40 anos

MISSÃO INSTITUCIONAL

Prestar assistência de excelência e referência com responsabilidade social, formar recursos humanos e gerar conhecimentos, atuando decisivamente na transformação de realidades e no desenvolvimento pleno da cidadania.

EDITAL N.º 03/2012 DE PROCESSOS SELETIVOS

PS 40 - ANALISTA DE DESENVOLVIMENTO DE TI I

Nome do Candidato: _____

Inscrição n.º: ___ ___ ___ ___ ___ ___ – ___

FAURGS
Fundação de Apoio da Universidade Federal do Rio Grande do Sul

SISTEMA DE GESTÃO DE QUALIDADE
FAURGS SETOR DE CONCURSOS
ISO 9001:2008

HOSPITAL DE CLÍNICAS
PORTO ALEGRE RS

HOSPITAL DE CLÍNICAS DE PORTO ALEGRE

EDITAL N.º 03/2012 DE PROCESSOS SELETIVOS

GABARITO APÓS RECURSOS

PROCESSO SELETIVO 40

ANALISTA DE DESENVOLVIMENTO DE TI I

01.	**C**	11.	**E**	21.	**A**	31.	**C**
02.	ANULADA	12.	**C**	22.	**E**	32.	**D**
03.	**C**	13.	**D**	23.	**C**	33.	**B**
04.	**A**	14.	**D**	24.	**E**	34.	**A**
05.	**A**	15.	**A**	25.	**A**	35.	**E**
06.	**D**	16.	**A**	26.	**B**	36.	**B**
07.	**B**	17.	**B**	27.	**C**	37.	**B**
08.	**B**	18.	**C**	28.	**E**	38.	**B**
09.	**D**	19.	**B**	29.	**D**	39.	**D**
10.	ANULADA	20.	**B**	30.	**E**	40.	**D**

HOSPITAL DE CLÍNICAS
PORTO ALEGRE RS

1971 • 2011 40 anos

INSTRUÇÕES

1. Verifique se este CADERNO DE QUESTÕES corresponde ao Processo Seletivo para o qual você está inscrito. Caso não corresponda, solicite ao Fiscal da sala que o substitua.

2. Esta PROVA consta de **40** (quarenta) questões objetivas.

3. Caso o CADERNO DE QUESTÕES esteja incompleto ou apresente qualquer defeito, solicite ao Fiscal da sala que o substitua.

4. Para cada questão objetiva, existe apenas **uma** (1) alternativa correta, a qual deverá ser assinalada na FOLHA DE RESPOSTAS.

5. **O candidato deverá responder à Prova Escrita, utilizando caneta esferográfica de tinta azul, fabricada em material transparente. Não será permitido o uso de lápis, lapiseira/grafite e/ou borracha e de caneta que não seja de material transparente durante a realização da Prova.** (conforme item 6.15.2 do Edital de Abertura)

6. Preencha com cuidado a FOLHA DE RESPOSTAS, evitando rasuras. Eventuais marcas feitas nessa FOLHA, a partir do número 41, serão desconsideradas.

7. Durante a prova, não será permitida ao candidato qualquer espécie de consulta a livros, códigos, revistas, folhetos ou anotações, nem será permitido o uso de telefone celular, transmissor/receptor de mensagem ou similares e calculadora.

8. Ao terminar a prova, entregue a FOLHA DE RESPOSTAS ao Fiscal da sala.

9. A duração da prova é de **3 (três) horas e 30 (trinta) minutos**, já incluído o tempo destinado ao preenchimento da FOLHA DE RESPOSTAS. Ao final desse prazo, a FOLHA DE RESPOSTAS será **imediatamente** recolhida.

10. **O candidato somente poderá se retirar da sala de Prova 1 (uma) hora após o seu início. Se quiser levar o Caderno de Questões da Prova Escrita Objetiva, o candidato somente poderá se retirar da sala de Prova 1 (uma) hora e 30 (trinta) minutos após o início.**

11. **O candidato que se retirar da sala de Prova, ao concluí-la, não poderá utilizar sanitários nas dependências do local de Prova.** (conforme item 6.15.7 do Edital de Abertura)

12. A desobediência a qualquer uma das recomendações constantes nas presentes instruções poderá implicar a anulação da prova do candidato.

Boa Prova!

01. O *Product Owner* é o responsável pelo gerenciamento do *backlog* do produto. O gerenciamento do *backlog* do produto **NÃO** inclui

(A) expressar claramente os itens do *backlog* do produto.
(B) ordenar os itens do *backlog* do produto para melhor alcançar as metas e missões.
(C) comunicar a visão, objetivo e itens do *backlog* do produto para a equipe de desenvolvimento.
(D) garantir o valor do trabalho realizado pelo time de desenvolvimento.
(E) garantir que a equipe de desenvolvimento entenda os itens do *backlog* do produto no nível necessário.

02. Na metodologia *Scrum*, durante a *sprint*

(A) não são feitas mudanças que podem afetar o objetivo da *sprint*.
(B) a composição da equipe de desenvolvimento pode ser modificada.
(C) as metas de qualidade diminuem.
(D) a equipe de desenvolvimento pode alterar o *backlog* do produto.
(E) as adaptações evolutivas podem ser negociadas com a equipe de desenvolvimento.

03. Assinale a alternativa que apresenta as fases do Processo Unificado.

(A) Modelagem de Negócio, Requisitos, Projeto
(B) Concepção, Projeto, Elaboração, Implantação
(C) Concepção, Elaboração, Construção, Transição
(D) Modelagem, Projeto, Elaboração, Construção, Implantação
(E) Análise de Requisitos, Elaboração, Construção, Transição, Implantação

04. Métodos de desenvolvimento ágil usualmente aplicam desenvolvimento iterativo e _____ de tempo _____, empregam planejamento _____, promovem entrega incremental e incluem outros valores e práticas que encorajam agilidade – resposta rápida e _____ à modificação.
Assinale a alternativa que preenche, correta e respectivamente, as lacunas do texto acima.

(A) evolutivo – limitado – adaptativo – flexível
(B) simplificado – ilimitado – adaptativo – flexível
(C) simplificado – limitado – adaptativo – flexível
(D) evolutivo – ilimitado – completo – flexível
(E) simplificado – limitado – com escopo fechado – inflexível

05. Como a maioria dos produtos de banco de dados relacional, o PostgreSQL suporta funções de agregação. Uma função de agregação computa um único resultado para várias linhas de entrada.
Assinale a alternativa que apresenta apenas funções de agregação do PostgreSQL.

(A) COUNT, SUM, AVG, MAX, MIN
(B) COUNT, UNION, MINUS, MAX, MIN
(C) UNION, MINUS, INTERSECTION, INNER JOIN, DISTINCT
(D) UNION, SELECT, JOIN, AVG, SUM
(E) COUNT, SUM, MINUS, MAX, MIN

06. No contexto do desenvolvimento de *software* orientado a objetos, considere as afirmações abaixo sobre acoplamento.

I - Uma subclasse é fortemente acoplada à sua superclasse.
II - O acoplamento forte favorece o projeto de classes independentes, o que reduz o impacto de modificações.
III- Classes com acoplamento forte são mais difíceis de reutilizar, pois seu uso requer a presença adicional de classes das quais são dependentes.

Quais estão corretas?

(A) Apenas I.
(B) Apenas III.
(C) Apenas I e II.
(D) Apenas I e III.
(E) I, II e III.

07. Para avaliar o comportamento interno do componente de *software*, os testes de *software* se utilizam da técnica denominada

(A) caixa-preta.
(B) caixa-branca.
(C) caixa-cinza.
(D) não funcional.
(E) regressão.

08. Considere o seguinte trecho de código:

```
public class DoException {
   public static void main (String args[]){
         try{
            doStuff();
         }
         catch (Exception exception){
               System.out.println("Handle in main method");
         }
   }
   public static void doStuff() throws Exception
   {
         try{
               System.out.println("Do some stuff");
               throw new Exception();
         }
         catch (RuntimeException runtimeException){
               System.out.println("Exception handle while doing some stuff");
         }
         finally{
               System.out.println("Finally message");
         }
   }
}
```

Quando compilarmos e rodarmos este código, o que é apresentado na console?

(A) Do some stuff
Handle in main method
NullPointerException

(B) Do some stuff
Finally message
Handle in main method

(C) Do some stuff
NullPointerException

(D) Do some stuff
Exception handle while doing some stuff
Finally message

(E) Erro em tempo de compilação

09. Considere o seguinte trecho de código:

```
public class Foo {
   int a = 0, b = 5;
   public static void main(String[] args) {
     Foo foo = new Foo();
     int a = 3, b = 8;
     foo.setValues(a, b);
     System.out.print(a + ":" + foo.a + ":" + foo.b);
   }
   void setValues(int a, int b) {
     a = 2;
     this.b = 3;
   }
}
```

O que é retornado à console ao compilarmos e executarmos este código?

(A) 3:3:8
(B) Falha em tempo de execução
(C) Falha em tempo de compilação
(D) 3:0:3
(E) 0:5:5

10. Assinale com **V** (verdadeiro) ou **F** (falso) as afirmações abaixo sobre conceitos de sobrecarga, sobrescrita e herança.

( ) O relacionamento É-UM é expresso em Java por meio das palavras-chave *extends* ou *implements*.
( ) O tipo retornado por um método não pode ser alterado quando este é sobrescrito.
( ) Ao ser sobrecarregado, o método pode retornar qualquer objeto que possa ser implicitamente convertido no tipo retornado pelo método original.
( ) Quando mudamos a assinatura de um método em uma subclasse, o tipo de retorno desse método pode ser alterado.
( ) Um método não pode retornar *null* quando deveria retornar referência a um objeto.

A sequência correta de preenchimento dos parênteses, de cima para baixo, é

(A) F – V – F – V – V.
(B) F – F – V – F – V.
(C) F – V – V – F – F.
(D) V – F – V – V – F.
(E) V – F – F – V – V.

11. Considere o seguinte trecho de código:

```
1 class veiculo {
2   void acelera() {
3           System.out.println("Veiculo acelerando");
4   }
5 }
6
7 class carro extends veiculo {
8   void acelera() {
9           System.out.println("Carro acelerando");
10          }
11 }
12
13 public class Garagem {
14          public static void main(String[] args) {
15                  veiculo v1 = new veiculo();
16                  carro c1 = new carro();
17                  carro c2 = (carro) v1;
18
19                  c1.acelera();
20                  c2.acelera();
21          }
22 }
```

Ao compilarmos e rodarmos este código, o que é apresentado na console?

(A) Carro acelerando
Carro acelerando
(B) Carro acelerando
Veículo acelerando
(C) Ocorre erro de compilação na linha 17
(D) Ocorre erro de compilação na linha 20
(E) Ocorre erro em tempo de execução

12. Sobre o tratamento de exceções em Java, é correto afirmar que

(A) na ocorrência de uma exceção, as variáveis declaradas em um bloco *try* podem ser acessadas no bloco *catch*, que irá realizar o tratamento dessa exceção.
(B) a ordem na declaração dos blocos *catch* não é importante, pois será executado somente o bloco que capturar a exceção.
(C) o bloco *finally* será sempre executado, mesmo que não sejam lançadas exceções pelo bloco *try* ou que seja executado um comando *return* pelo bloco *try*.
(D) caso uma exceção seja lançada dentro de um bloco *catch*, o fluxo de execução é retornado ao método que executou o método onde ocorreu a exceção, sem que seja executado o bloco *finally*.
(E) instâncias da classe *RuntimeException* representam falhas ou erros que podem ocorrer em tempo de execução; são exceções do tipo verificadas, sendo que o compilador exige que sejam capturadas ou lançadas à classe que chamou o método.

13. Considere o seguinte trecho de código:

```
public class a2D {
   public static void main(String[] args) {
     int[][] a2d = new int[5][6];
     int valor = 1;
     for (int i = 0; i < a2d.length; i++) {
              for (int j = 0; j < a2d[i].length; j++) {
                       a2d[i][j] = valor;
                       valor += 1;
              }
     }
     System.out.print(a2d[1][1]);
   }
}
```

Ao copilarmos e rodarmos este código, o que é apresentado na console?

(A) 1
(B) 2
(C) 7
(D) 8
(E) 11

14. Considere as afirmações abaixo sobre a IDE Eclipse.

I - Possui diversos recursos que auxiliam no processo de desenvolvimento, como editor de código-fonte, gerador de código, compilador, depurador e executor de testes automáticos. O Eclipse pode ser usado no desenvolvimento de *software* nas linguagens Java, C++ e PHP.

II - Além do desenvolvimento de *software*, também possui um console que pode ser utilizado na execução de programas.

III- Sendo uma IDE livre e gratuita, o Eclipse possui algumas limitações. Uma das dificuldades que podemos destacar é a falta de integração com sistemas de controle de versão, como SVN e CVS.

Quais estão corretas?

(A) Apenas I.
(B) Apenas II.
(C) Apenas III.
(D) Apenas I e II.
(E) I, II e III.

15. Sistemas de controle de versão são convenientes quando diversos desenvolvedores trabalham sobre o mesmo projeto simultaneamente, resolvendo eventuais conflitos entre as alterações. Sobre sistema *SubVersion* é **INCORRETO** afirmar:

(A) implementa apenas o conceito *lock-modify-unlock*, pois este modelo impede a ocorrência de conflitos por arquivos terem sido modificados simultaneamente por diferentes usuários.

(B) cada desenvolvedor pode ter, em sua máquina local, mais de uma versão do mesmo arquivo.

(C) *checkout* é usado para denominar o *download* inicial de um módulo inteiro a partir do repositório.

(D) cada vez que é realizado *commit* de um arquivo modificado localmente, o sistema de controle de versão atribui a esse arquivo uma numeração chamada *revision.*

(E) permite a criação de *branches*, usados para uma divisão dos arquivos de um projeto em linhas de desenvolvimento independentes. *Branches* são úteis no teste de uma nova funcionalidade ou em projetos destinados a um cliente específico.

16. Considere o código abaixo.

```
public class Xy {
   public static void main(String args[]) {

     String x = new String("abc");
     String y = x;

     x.concat(" e mais um pouco");
     x = x.concat(" no final");

     System.out.println("x = " + x);
     System.out.println("y = " + y);
   }
}
```

O que é impresso com a compilação e execução deste código?

(A) x = abc no final
    y = abc

(B) x = abc e mais um pouco no final
    y = abc e mais um pouco no final

(C) x = abc no final
    y = abc no final

(D) x = abc e mais um pouco no final
    y = abc

(E) x = no final
    y = abc

17. Uma tabela de dispersão, também conhecida por tabela de espalhamento ou *Hashtable*, é uma estrutura de dados especial que associa chaves de pesquisa a valores. A linguagem Java oferece as estruturas de coleções *HashMap* e *HashSet* que usam o valor *HashCode* de um objeto para o armazenamento e busca nessas estruturas.
Segundo os contratos definidos na API Java para a sobrescrita dos métodos *hashCode*() e *equals*(), podemos afirmar que

(A) se *x.equals*(y) retornar *false*, *x.hashCode*() == *y.hashCode*() retornando *true*, indica maior eficiência no uso da estrutura de dados.
(B) se *x.equals*(z) e *y.equals*(z) retornarem *true*, então *z.equals*(x) retornará *true*.
(C) caso não ocorra a sobrescrita do método *equals*(), os objetos dessa classe serão chaves úteis de *hashing*.
(D) se *x.equals*(y) retornar *true*, *x.hashCode*() e *y.hashCode*() não terão o mesmo resultado.
(E) após um objeto ser instanciado, a execução do seu método *hashCode*() retornará sempre o mesmo valor, independentemente das informações usadas pelo método *equals*().

18. O conceito de conversação introduzido pelo *Seam* é um novo escopo do contexto em tempo de execução. Sobre esse conceito, podemos afirmar que

(A) uma conversação só é finalizada com *links* de navegação ou na execução de um método anotado com *@Finalize*.
(B) um usuário utiliza uma única conversação. Ela é criada no início da sua sessão e termina quando o usuário finaliza a aplicação.
(C) cada conversação ativa possui o seu próprio estado e conjunto de dados. O *Seam* pode iniciar ou encerrar cada conversação separadamente.
(D) uma conversação é criada com uma notação *@Begin* em um método *action listener*, mesmo quando esse método não é executado com sucesso ou finalize com uma exceção.
(E) se uma conversação explícita ainda estiver ativa, o *Seam* trata uma nova notação *@Begin,* promovendo a conversação já existente para uma conversação de longa duração.

19. Considere o código abaixo.

```
class Animal {
        void makeNoise() {
                System.out.println("Generic Noise");
        }
}
class Dog extends Animal {
        void makeNoise() {
                System.out.println("Hauf");
        }
        void playDead() {
                System.out.println("Roll Over");
        }
}
public class CastTest {
        public static void main(String[] args) {
                Animal[] a = { new Animal(), new Dog(),
            new Animal() };

                for (Animal animal : a) {
                        _____________________

                        ((Dog) animal).playDead();
                }
        }
}
```

Assinale a alternativa cujo código completa a lacuna, fazendo com que a classe *CastTest* compile e seja executada sem erros.

(A) animal = new Animal();
(B) if(animal instanceof Dog)
(C) if(animal extends Dog)
(D) if(animal.equals(Dog))
(E) Dog a = new Dog();

20. Um *framework* de sistemas é um conjunto de funcionalidades genéricas e personalizáveis. Sua estrutura consiste em uma plataforma de *software* reutilizável e universal no desenvolvimento de aplicativos, produtos e soluções. Considere as seguintes afirmações sobre alguns dos *frameworks* mais utilizados com Java.

I - O *Java Pages Assistant*, também conhecido como JPA, é um conjunto de bibliotecas fornecido pela *Oracle* para facilitar o desenvolvimento de páginas XHTML. Ele provê um conjunto de *tags* e componentes que facilitam a integração entre a página e *Java beans*, validadores, conversores e outros objetos que estão alocados no servidor de aplicação.

II - A sigla AJAX é um acrônimo para *Asynchronous Java eXchange*, um protocolo de comunicação entre aplicações por meio de mensagens. Uma vez recebidas, essas mensagens são armazenadas em filas no servidor de aplicação e processadas de forma assíncrona.

III- *Hibernate* é um dos *frameworks* utilizados para facilitar o armazenamento e a consulta em bases de dados. Ele permite ao desenvolvedor utilizar o estilo de modelos de domínio estilo POJO (*Plain Old Java Object*) em suas aplicações, de forma a estender o conceito de mapeamento objeto-relacional.

Quais estão corretas?

(A) Apenas I.
(B) Apenas III.
(C) Apenas I e III.
(D) Apenas II e III.
(E) I, II e III.

21. Existem diversas formas de se enviar requisições AJAX em uma página JSF. Nesse contexto, imagine-se utilizando as ferramentas fornecidas pelo *framework Richfaces*. Quando se deseja programar o envio de uma requisição AJAX, utilizando um temporizador, a *tag* mais indicada é:

(A) <a4j:poll>
(B) <a4j:support>
(C) <a4j:loadBundle>
(D) <a4j:ajaxListener>
(E) <a4j:commandLink>

22. Um _____ é um servidor que disponibiliza às _____ serviços como segurança, suporte a transações, balanceamento de carga, entre outros. Como exemplos de servidores que suportam _____, podemos citar o _____ e o _____.
Assinale a alternativa que preenche, correta e respectivamente, as lacunas do texto acima.

(A) servidor *web* – páginas *web* – PHP – *Apache* HTTP *Server* – *Oracle GlassFish*
(B) servidor de aplicações – aplicações de *software* – .NET – Microsoft .NET Framework – IBM WebSphere Application Server
(C) servidor de banco de dados – aplicações de *software* – .NET – Microsoft Active Directory – IBM DB2
(D) servidor de arquivos – empresas – EXT3 – Microsoft Active Directory – OpenLDAP
(E) servidor de aplicações – aplicações de *software* – Java – Oracle WebLogic Application Server – IBM WebSphere Application Server

23. Considere uma tecnologia – um *framework* de componentes no lado do servidor para construir aplicações *web* baseadas em Java – que consiste de uma API para representar componentes e gerenciar o seu estado, controlar eventos, validar e converter dados no lado do servidor, definir navegação entre páginas, suportar internacionalização e acessibilidade e prover extensibilidade para essas ações. O *framework* consiste ainda de *Tag libraries* para adicionar componentes em páginas *web* e para ligar componentes a objetos no lado do servidor. Assinale a alternativa que apresenta a tecnologia em questão.

(A) XHTML
(B) JavaServer Pages
(C) JavaServer Faces
(D) Java Pages Assistant
(E) Java Runtime Environment

24. Existem *tags* que representam componentes HTML para receber entrada de dados ou mostrar dados aos usuários. Esses dados são coletados como partes de um formulário e submetidos ao servidor, geralmente quando o usuário clica em um botão.
Páginas *web* representam a camada de apresentação em aplicações *web*. O processo de criar páginas *web* para uma aplicação Java *web* inclui adição de componentes para a página e a ligação a *beans* gerenciados, validadores, *listeners,* conversores e outros objetos do lado do servidor associados com a página. Neste contexto, considere as seguintes afirmações.

I - Conversores são usados para converter dados recebidos dos componentes de entrada.

II - *Listeners* são usados para ouvir os eventos que acontecem na página e realizar ações conforme definido.

III- Validadores são usados para checar se os dados recebidos dos componentes de entrada são válidos.

Quais estão corretas?

(A) Apenas III.
(B) Apenas I e II.
(C) Apenas I e III.
(D) Apenas II e III.
(E) I, II e III.

25. Considere as afirmações sobre a *tag* do *RichFaces* <a4j:support> - org.ajax4jsf.taglib.html.jsp.AjaxSupport.

I - A *tag* <a4j:support> é a mais importante na biblioteca *RichFaces.* Ela propicia a componentes comuns terem funcionalidades *Ajax*. Os outros componentes *Ajax RichFaces* são baseados nos mesmos princípios que a <a4j:support> possui.

II - A *tag* <a4j:support> deve ser usada dentro de outros componentes, como o <h:selectOneMenu>, mas apresenta a limitação de não poder ser usada em campos do tipo entrada de texto, como <h:inputText>.

III- A *tag* tem dois atributos principais: "*event*", que define o evento *JavaScript* ao qual o suporte *Ajax* estará ligado; "*focus*", que diz quais componentes JSF devem ser renderizados novamente após uma requisição *Ajax*.

Quais estão corretas?

(A) Apenas I.
(B) Apenas II.
(C) Apenas III.
(D) Apenas I e II.
(E) Apenas I e III.

26. Entre os componentes *RichFaces* existe um que, à medida que são digitados caracteres em uma caixa de texto, apresenta uma lista de opções ao usuário. A partir dessa lista, o usuário escolhe um item e a tela executa uma determinada ação. As figuras abaixo ilustram esse comportamento.

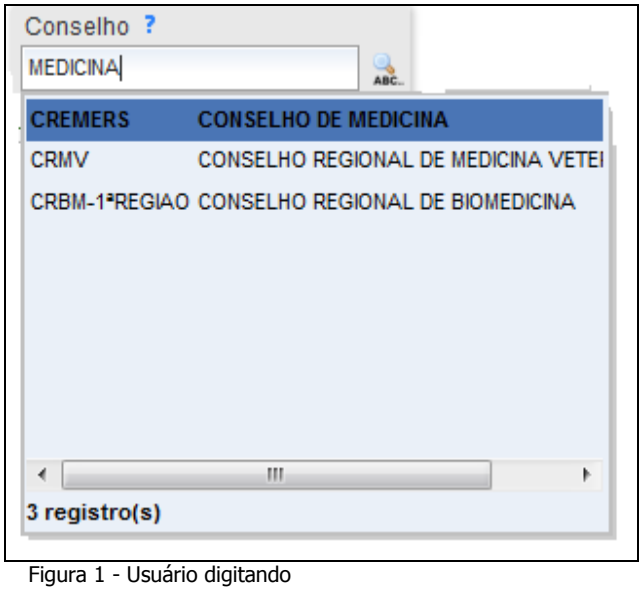

Figura 1 - Usuário digitando

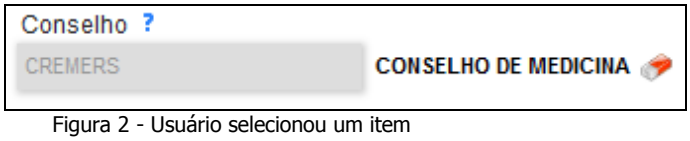

Figura 2 - Usuário selecionou um item

Assinale a alternativa que apresenta o componente em questão.

(A) <rich:inputList>
(B) <rich:suggestionBox>
(C) <rich:suggestionList>
(D) <rich:inputSuggestionBox>
(E) <rich:inputSuggestionList>

27. Considere a seguinte tabela em um banco de dados Oracle.

FUNCIONARIO			
COLUMN_NAME	**DATA_TYPE**	**NULLABLE**	**PRIMARY KEY**
ID	NUMBER	No	TRUE
MATRICULA	NUMBER	No	FALSE
CONJUNTO	VARCHAR2(80)	No	FALSE
CRIADO_EM	DATE	Yes	FALSE

Figura 1 - Tabela Funcionário

Você é encarregado de efetuar o mapeamento objeto-relacional dessa tabela em uma aplicação Java. Entre as alternativas a seguir, assinale a que **NÃO** corresponde às melhores práticas de desenvolvimento e que, se implementada, acarreta comportamento indesejável na aplicação.

(A) a classe deve ter a notação @javax.persistence.Entity
(B) a coluna "ID" deve ter a notação @javax.persistence.Id
(C) a coluna "CONJUNTO" deve ser mapeada como um array de Bytes
(D) a coluna "CRIADO_EM" pode ser mapeada como do tipo java.util.Date
(E) a coluna "MATRICULA" pode ser mapeada como do tipo java.lang.Long

28. A *API EntityManager* permite trocar o estado de uma entidade, ou seja, carregar e armazenar objetos. Numere a segunda coluna de acordo com a primeira, associando os estados de uma entidade com o seu significado.

(1) *New* (transiente)
(2) *Managed* (persistente)
(3) *Detached*
(4) *Removed*

( ) Entidade cuja instância tem uma identidade persistente e está associada a um contexto persistente.
( ) Entidade cuja instância tem uma identidade persistente, que não mais está associada com um contexto persistente.
( ) Entidade cuja instância tem uma identidade persistente, associada com um contexto persistente, mas agendada para ser apagada do banco de dados.
( ) Entidade recém-instanciada, ainda não associada a um contexto de persistência. Ela não está presente no banco de dados e nenhum valor identificador foi associado a ela.

A sequência numérica correta de preenchimento dos parênteses da segunda coluna, de cima para baixo, é

(A) 3 – 2 – 4 – 1.
(B) 2 – 1 – 4 – 3.
(C) 3 – 2 – 1 – 4.
(D) 1 – 3 – 2 – 4.
(E) 2 – 3 – 4 – 1.

29. Suponha que você está trabalhando em uma aplicação JAVA *web*, na IDE Eclipse. No seu ambiente de trabalho, todas as configurações estão corretas, as bibliotecas e dependências definidas. Existe uma classe que mapeia uma tabela do banco de dados chamada Servidor que, entre outros, possui um campo "nome" definido como java.lang.String. Você é solicitado a realizar uma consulta nessa tabela pelo campo "nome", sem diferenciar maiúsculas de minúsculas; o parâmetro passado na pesquisa pode estar no início, no meio, no fim ou ser o valor exato do campo no banco de dados.

Qual dos seguintes trechos de código, usando a DetachedCriteria do *Hibernate*, é o mais apropriado para essa situação?

(A)
```
DetachedCriteria criteria = DetachedCriteria.forClass(Servidor.class);
criteria.add(Restrictions.like("nome", stParametro, MatchMode.ANYWHERE));
```

(B)
```
DetachedCriteria criteria = DetachedCriteria.forClass(Servidor.class);
criteria.add(Restrictions.like("nome", stParametro, new MatchMode[] { MatchMode.START, MatchMode.END, MatchMode.EXACT}, CaseSensitive.FALSE));
```

(C)
```
DetachedCriteria criteria = DetachedCriteria.forClass(Servidor.class);
criteria.add(Restrictions.ilike("nome", stParametro, new MatchMode[] { MatchMode.START, MatchMode.END, MatchMode.EXACT}));
```

(D)
```
DetachedCriteria criteria = DetachedCriteria.forClass(Servidor.class);
criteria.add(Restrictions.ilike("nome", stParametro, MatchMode.ANYWHERE));
```

(E)
```
DetachedCriteria criteria = DetachedCriteria.forClass(Servidor.class);
criteria.add(Restrictions.like("nome", stParametro, MatchMode.ANYWHERE,
CaseSensitive.FALSE));
```

30. Java EE 5 introduz o conceito de contexto de persistência estendido. Com relação aos benefícios desse conceito, considere as afirmações abaixo.

I - Permite carregamento "*lazy*" das associações entre entidades e "*proxies*" não inicializados.

II - Previne entidades desatachadas.

III- Trabalha em conjunto com "*lock*" otimista para auxiliar no uso de unidades de trabalho mais longas.

Quais estão corretas?

(A) Apenas I.
(B) Apenas II.
(C) Apenas I e II.
(D) Apenas II e III.
(E) I, II e III.

31. *TDD - Test Driven Development* é uma técnica de _____ com a qual, em pequenas iterações, são desenvolvidos testes automatizados que definem requisitos em código, onde primeiro se escreve _____ e depois _____ da aplicação. Cada iteração deve começar com um teste que falhe e terminar com todos os testes executando com sucesso.

(A) gerenciamento de projeto – a documentação – o código
(B) gerenciamento de projeto ágil – os casos de uso – a implementação
(C) desenvolvimento de *software* – o teste – o código
(D) teste de *software* – a interface – a implementação
(E) teste de *software* – a implementação – a interface

32. Numere a segunda coluna de acordo com a primeira, associando os conceitos de desenvolvimento *de software* às suas respectivas descrições.

(1) Coesão
(2) Acoplamento
(3) Projeto modular
(4) *MVC - Model-View-Controller*
(5) Modelo de domínio

( ) Medida de quanto um elemento está conectado, tem conhecimento ou depende de outros elementos.
( ) Medida de quanto as responsabilidades de um elemento estão fortemente relacionadas e focalizadas.
( ) Propriedade de um sistema que foi decomposto em conjuntos de elementos coesos e fracamente acoplados.
( ) Representação de classes conceituais do mundo real em um determinado contexto.
( ) Arquitetura ou padrão de projeto que divide as funcionalidades do sistema em camadas.

(A) 1 – 2 – 4 – 3 – 5
(B) 2 – 1 – 5 – 4 – 3
(C) 4 – 2 – 1 – 3 – 5
(D) 2 – 1 – 3 – 5 – 4
(E) 2 – 1 – 5 – 3 – 4

33. O conceito em Orientação a Objetos que promove a construção de *software* flexível e reutilizável, através da capacidade dos objetos serem substituíveis por outros com interface coincidentes, é

(A) herança.
(B) polimorfismo.
(C) associação.
(D) generalização.
(E) acoplamento.

34. O *Jboss Seam* utiliza anotações no código para definir componentes, injeções de componentes, segurança, etc. Para definir uma classe como um componente *Seam* em escopo *default* de aplicação, são utilizadas as seguintes anotações:

(A) @Name("Componente1") e @Scope(ScopeType.APPLICATION) na declaração da classe
(B) @Startup na declaração da propriedade que recebe injeção do componente
(C) @Scope() na declaração da classe
(D) @Install(scope="application") na declaração da propriedade que ejeta a classe
(E) @Name(name="Component1",scope=" ScopeType .APPLICATION") na declaração da classe

35. Para injetar um componente *Seam* em escopo de aplicação, na propriedade de uma classe, usamos

(A) a anotação *@In* na propriedade ou no método "*set*" da propriedade, independentemente se a classe é ou não um componente *Seam*.
(B) a anotação *@Out*.
(C) a anotação *@InjectComponent* na propriedade.
(D) somente o arquivo *components.xml*.
(E) a anotação *@In* na propriedade ou no método "*set*" da propriedade, desde que a classe também seja um componente *Seam*.

36. A chave para a aplicação do padrão de projeto _____ é projetar interfaces genéricas o bastante para suportar uma variedade de algoritmos para a estratégia e seu contexto.

Assinale a alternativa que completa, adequadamente, a lacuna do texto acima.

(A) *Observer*
(B) *Strategy*
(C) *Builder*
(D) *Singleton*
(E) *Proxy*

37. Considere as afirmações abaixo.

I - Programar para uma interface, não para uma implementação.
II - Os elementos de um *software* colaboram entre si, baseando-se em obrigações e benefícios mútuos.
III- Primeiramente deve-se escrever a implementação e depois os testes manuais.

Quais são princípios de projetos orientados a objetos?

(A) Apenas I.
(B) Apenas I e II.
(C) Apenas I e III.
(D) Apenas II e III.
(E) I, II e III.

38. Refatoração é uma _____ disciplinada para reestruturação de um código existente, alterando sua estrutura interna _____. Cada transformação deverá ser _____, mas uma sequência de transformações pode produzir uma reestruturação significante. O sistema é mantido em pleno funcionamento após cada refatoração, reduzindo as chances do sistema ficar seriamente quebrado durante a reestruturação.

Assinale a alternativa que completa, correta e respectivamente, as lacunas do texto acima.

(A) metodologia de gerenciamento de projetos – sem modificar seu comportamento externo – grande
(B) técnica – sem modificar seu comportamento externo – pequena
(C) técnica – e seu comportamento externo – pequena
(D) técnica – e seu comportamento externo – grande
(E) análise – sem modificar seu comportamento – pequena

39. Assinale a alternativa que apresenta práticas consideradas ágeis.

(A) Refatoração – Método cascata – Testes unitários
(B) TDD – Curva ABC – Integração contínua
(C) Documentação extensa e detalhada no início do projeto – Programação em par – Entregas frequentes
(D) Integração contínua – Entregas frequentes – Testes unitários
(E) Pirâmide de Maslow – Entregas frequentes – Cartões CRC

40. O padrão de projeto _____ define o esqueleto de um algoritmo, postergando a definição de alguns passos para subclasses, permitindo que as subclasses redefinam certos passos do algoritmo sem mudar sua estrutura.

Assinale a alternativa que completa, adequadamente, a lacuna do texto acima.

(A) *Observer*
(B) *Strategy*
(C) *Builder*
(D) *Template Method*
(E) *Proxy*