CONHECIMENTOS ESPECÍFICOS

» PROGRAMAÇÃO (Perfil 02) «

21. Analise as seguintes sentenças, considerando o código escrito na linguagem PHP 5 e ilustrado pelo quadro abaixo:

```
1     <%php
2           $s1 = array ( “candidato 1”, 3=>”candidado 2”,”candidato 3”);
3           $s2 = “x15”;
4           $s3 = (int) $s2;
5           $s4 = “s5”;
6           $s5 = 20;
7           echo $$s4;
8           $s6 = “20”;
9           if  ($s5 === $s6)
10                echo $s6;
11    %>
```

I. As chaves geradas para o *array* $s1 são 1, 3 e 2.
II. O valor atribuído a $s3 será o inteiro 15.
III. $$s4 representa uma variável dinâmica, logo, teremos como saída, na linha 7, o valor inteiro 20.
IV. A comparação realizada na linha 9 retornará verdadeiro.

Sobre as sentenças anteriores, está (ão) CORRETAS(S):

a) Apenas a sentença I.
b) Apenas a sentença II.
c) Apenas a sentença III.
d) Apenas a sentença IV.
e) Apenas as sentenças I e II.

22. Referente à linguagem PHP 5, é INCORRETO afirmar:

a) Em PHP 5 é possível realizar engenharia reversa completa, em qualquer classe, através de um conjunto de classes presentes no módulo de reflexão.
b) As variáveis contendo informações relativas ao servidor web estão contidas em uma variável superglobal.
c) O SimpleXML provê um conjunto de ferramentas que permitem converter o conteúdo de um XML em objetos.
d) Um dos padrões utilizados nas funções que tratam com expressões regulares é o padrão POSIX (1003.2).
e) Diferentemente da linguagem Java, PHP 5 não permite o uso de drives ODBC para conexões com bancos de dados.

23. Classes *sockets* em Java são utilizadas para representar uma conexão entre programas cliente e servidor. Sobre *sockets* em Java, é CORRETO afirmar:

a) A classe "ServerSocket" implementa as conexões tanto do lado cliente quanto do lado servidor.

b) A classe "Socket" permite prover o uso do protocolo Transport Layer Security (TLS).

c) A "DatagramPacket" é uma classe que representa um socket para enviar e receber pacotes datagrama.

d) Através da classe "SocketPermission", é possível especificar um conjunto de ações a um host.

e) A classe "MulticastSocket" permite o envio de pacotes multicast para qualquer classe de endereços IP.

24. Considerando a Java RMI, assinale a alternativa CORRETA:

a) O cliente invoca um método no skeleton local, o qual é responsável pela invocação do método no objeto remoto.

b) RMI itera com sistemas existentes através da interface de métodos nativos JNI.

c) Os objetos remotos só podem implementar uma interface remota.

d) RMI faz uso do protocolo JRMP para comunicação com aplicações clientes CORBA.

e) RMI não prevê canais seguros entre cliente e servidor.

25. Associe as duas colunas abaixo em relação a Java RMI.

I. java.rmi.Naming

II. java.rmi.registry.LocateRegistry

III. java.rmi.server.UnicastRemoteObject

IV. java.rmi.activation.Activatable

1. Fornece métodos para armazenar objetos remotos em um registro de objetos remotos.
2. Fornece suporte a objetos remotos que requerem acesso persistente ao longo do tempo.
3. Exporta um objeto remoto com JRMP.
4. Permite criar um registro de objetos remotos.
5. Obtém um *stub* para comunicação com o objeto remoto.

A sequência CORRETA dessa associação é:

a) I-1; II-4; III-3; III-5; IV-2

b) I-4; II-1; III-5; IV-2; IV-3

c) I-1; II-4; III-3; III-2; IV-5

d) I-4; II-1; II-3; III-5; IV-2

e) I-1; I-4; II-5; III-2; IV-3

26. Analise as seguintes sentenças sobre *Beans* de Sessão.

I. A anotação @javax.ejb.EJB serve apenas para injetar uma instância de um Bean de Sessão junto a um atributo na classe.
II. Para referenciar múltiplos EJBs, podemos utilizar a anotação: @javax.ejb.EJBs.
III. A anotação @javax.annotation.Resource tem como uma das funções injetar recursos de conexões ao banco de dados definidos no arquivo persistence.xml.

Sobre as sentenças anteriores, está (ão) CORRETAS(S):

a) Apenas as sentenças I e II.
b) I, II e III.
c) Apenas as sentenças II e III.
d) Apenas a sentença I.
e) Apenas a sentença II.

27. Exceto durante a invocação do método, todas as suas instâncias são equivalentes, permitindo que o *container* EJB atribua uma instância para qualquer cliente. Essa afirmação refere-se ao:

a) *Stateful Session Bean*
b) *Singleton Session Bean*
c) *Stateless Session Bean*
d) Interceptador
e) *Container* EJB

28. Um *bean* de sessão *Singleton* é apropriado em várias circunstâncias, EXCETO, quando

a) o *bean* fizer a mediação entre o cliente e os outros componentes da aplicação, apresentando uma visão simplificada para o cliente.
b) for necessário o estado do *bean* ser compartilhado entre a aplicação.
c) um simples *enterprise bean* necessitar de ser acessado por múltiplas *threads* concorrentes.
d) a aplicação necessitar de um *enterprise bean* para desempenhar tarefas na inicialização e finalização da aplicação.
e) o *bean* implementar um *web service*.

29. Analise o quadro a seguir.

```
A (activationConfig={
                @ActivationConfigProperty(
                                propertyName = “destinationType”,
                                propertyValue = B)})
public class ProcessadorReservaBean implements C
{
        public void D (Message message) {
                …
        }
}
```

Para que o código ilustrado pelo quadro tenha a estrutura de um *bean enterprise* baseado em mensagem, o modelo de mensagem seja ponto-a-ponto, as letras **A**, **B**, **C** e **D** poderão ser substituídas, respectivamente, por:

a) @MessageDriven, “javax.jms.P2p”, MessageListener, onReceive

b) @MessageContext, “javax.jms.Queue” , MessageDrivenContext, send

c) @MessageDriven, “javax.jms.Topic”, Message, onMessage

d) @MessageDriven, “javax.jms.Queue”, MessageListener, onMessage

e) @Message, “javax.jms.P2p”, Message, onReceive

30. Considere as seguintes sentenças sobre a API de Timer Service de EJB.

I. Um temporizador pode ser uma classe que implementa “javax.ejb.TimedObject”, possuindo um método de assinatura “void ejbTimeout(javax.ejb.Timer)”.
II. Um temporizador pode ser um Bean de Sessão Stateless que possui qualquer método anotado com @Time e um parâmetro do tipo “javax.ejb.Timer”.
III. Temporizadores podem ser executados uma ou várias vezes em um determinado intervalo de tempo.

Sobre as sentenças anteriores, está (ão) CORRETAS(S):

a) Apenas a sentença II.

b) Apenas a sentença III.

c) Apenas as sentenças II e III.

d) Apenas as sentenças I e II.

e) Apenas as sentenças I e III.

31. Sobre os métodos “getReference” e “find” da interface javax.persistence.EntityManager, é FALSO afirmar que:

a) Ambos os métodos retornam objetos do mesmo tipo.

b) Ambos possuem como finalidade, retornar uma instância de alguma entidade.

c) O método find possui sobrecargas.

d) Ambos retornam null caso a entidade não seja encontrada.

e) O método getReference não possui sobrecargas.

32. Na especificação de *Servlets* 3.0, uma classe que implementa a interface HttpSessionBindingListener, deve implementar os métodos:

a) valueInit e valueDestroy

b) valueBound e valueUnbound

c) valueCreated e valueDestroyed

d) valueInitialized e valueDestroyed

e) boundValue e unboundValue

33. De acordo com a especificação de *Servlets* 3.0, sobre parâmetros, pode-se afirmar que:

a) Estes só podem ser obtidos através dos métodos getParameter e getInitParameter.

b) Os métodos que retornam parâmetros têm como retorno o tipo *Object*.

c) Estes podem ser modificados através do método setParameter.

d) Os métodos de recuperação de parâmetros existem nas interfaces ServletContext, ServletConfig e ServletRequest.

e) Os parâmetros não podem ser declarados no web.xml.

34. A fase do ciclo de vida do JSF (*JavaServer Faces*) em que os valores dos componentes são atualizados de acordo com os valores recebidos na requisição (usando conversores, caso existam) é:

a) *Apply Request Values*

b) *Invoke Application*

c) *Render Response*

d) *Update Model Values*

e) *Process Validations*

35. Em JSF podemos criar conversores personalizados implementando a interface *Converter*. As classes que implementam *Converter* devem possuir os métodos

a) convert e convertTo

b) setObject e setString

c) setAsObject e setAsString

d) getObject e getString

e) getAsObject e getAsString

36. Sobre SimpleTags, é CORRETO afirmar que:

a) cada SimpleTag deve estender a classe TagSupport.

b) para cada atributo da *tag*, deve existir um método set, no estilo JavaBeans, para que o valor do atributo seja lido pela classe que representa o *handler* da *tag*.

c) uma SimpleTag vazia só deve ser chamada na forma <x/>.

d) usamos o método *getTagBody* para obter o corpo da *tag* e poder manipulá-lo dentro da classe que representa o *handler*.

e) as SimpleTags necessitam de uma classe que represente um *handler* e contenha o método *doBody*.

37. Os princípios abaixo encorajam aplicações RESTful a serem simples, leves e rápidas, EXCETO:

a) Interface uniforme

b) Mensagens auto-descritivas

c) Interações stateful através de hiperlinks

d) Identificação de recursos através de URI

e) Estados representacionais definidos apenas pela W3C

38. A JAX-RS é uma API projetada para tornar fácil o desenvolvimento de aplicações que usam a arquitetura REST.

```
@Path("/meuRecurso")
@Produces("text/plain")
public class AlgumRecurso {

    @POST
    @Produces("application/xml")
    public String doPost(FormURLEncodedProperties formData) {
        ...
    }

    @POST
    @Path("/{cpf}")
    public String doPost2(String cpf) {
        ...
    }
}
```

Considerando o código fonte ilustrado pelo quadro acima e que este utiliza a JAX-RS, assinale a alternativa INCORRETA

a) Se, em um recurso, nenhum método for capaz de produzir o tipo MIME pedido por um cliente, o *runtime* JAX-RS enviará ao cliente a mensagem HTTP "406 Not Acceptable".

b) Para que o valor do “cpf” seja extraído da URI do recurso “doPost2”, o parâmetro desse método deverá ser redefinido para (@QueryParam(“cpf”) String cpf).

c) O método “doPost” produzirá o MIME “application/xml” e não o MIME "text/plain" definido no nível da classe.

d) A JAX-RS permite que o “cpf”, passado através da URI do “doPost2”, seja validado por uma expressão regular.

e) O *runtime* JAX-RS poderá retornar ao cliente um documento WADL descrevendo o recurso “AlgumRecurso”.

39. A JAX-WS é uma API pertencente à JEE, que provê funcionalidades para o desenvolvimento de *web services*. Sobre essa API, é INCORRETO afirmar:

a) Em um serviço JAX-WS é possível processar uma mensagem SOAP sem a publicação de um WSDL.

b) A JAX-WS permite aos desenvolvedores a criação de *web services* orientados a RPC (*Remote Procedure Call*).

c) O pacote javax.jws oferece uma API para a manipulação de *web services* publicados em uma UDDI.

d) O sistema *runtime* do JAX-WS converte as mensagens SOAP trocadas entre o produtor e o consumidor do *web service*.

e) A *Service Endpoint Implementation* (SEI) é uma classe Java que declara os métodos pelos quais o cliente pode invocar no *web service*.

40. Considerando o código escrito utilizando a JAX-WS ilustrado no quadro abaixo.

```
@Stateless
@WebService (name=”AgenteViagem”)
public class AgenteViagemBean implements AgenteViagemRemote {
        @WebMethod (operationName = “Reserva”)
        public String reservar (int clienteID, int cabineID, double preco) {
                        …
        }
}
```

Analise as sentenças abaixo:

I. O atributo “operationName” é utilizado para definir a operação WSDL que o método “reservar” implementa.
II. Os parâmetros do método “reservar” estarão configurados, na WSDL, com os “wsld:part” nomeados de “clienteID”, “cabineID” e “preco”.
III. Na WSDL, o *web service* terá o valor “AgenteViagem” como o nome do “portType”.
IV. O componente “AgenteViagemBean” só poderá ser acessado, remotamente, por meio de um *web service*, devido a inclusão da anotação @WebService.

Sobre as sentenças anteriores, está (ão) CORRETAS(S):

a) Apenas a sentença I.
b) Apenas a sentença II.
c) Apenas as sentenças I e III.
d) Apenas as sentenças II e III.
e) I, II, III e IV.

41. Considerando o desenvolvimento de uma aplicação com *Rails*, analise as sentenças abaixo:

I. O *Active Record* é a camada ORM (*Object-Relational Mapping*) fornecida com *Rails*.
II. Os métodos *callbacks* podem poluir um modelo com códigos que não estão diretamente relacionados ao seu propósito.
III. O controlador *Rails* gerencia sessões e *caching*.
IV. *Rails* abstrai *cookies* atrás de uma interface que permite a serialização de objetos.

Sobre as sentenças acima está(ão) CORRETA(S):

a) Apenas a sentença I.
b) Apenas as sentenças I e II.
c) Apenas as sentenças II e III.
d) Apenas as sentenças I, II e III.
e) I, II, III e IV.

42. O componente **X** provê um *framework* para gerenciar a conexão entre objetos de negócio e *web services Restful*. A que componente *Rails*, **X** se refere?

a) *Active Record*

b) *Active Resource*

c) *Active Service*

d) *Active Support*

e) *Active Dispatch*

43. Considerando o *Active Record*, referente ao relacionamento entre tabelas, assinale a alternativa CORRETA:

a) A herança *single-table* define a hierarquia necessária ao modelo de classes, mas não garante que uma tabela corresponda à classe base de uma hierarquia.

b) O *Active Record* provê suporte para a organização de linhas de uma tabela dentro de uma estrutura hierárquica. Esse suporte é conhecido como o modelo "*Acts As List"*.

c) Associações polimórficas ligam objetos de diferentes tipos, pressupondo que estes compartilham características comuns, mas que esses objetos terão representações diferentes.

d) Para expressar uma relação N:M (muitos-para-muitos) entre dois modelos, adiciona-se a declaração "many_to_many" em ambos os modelos.

e) "has_one" define um atributo que se comporta como uma coleção de objetos filhos.

44. Em relação ao arquivo AndroidManifest.xml, qual das alternativas abaixo é FALSA?

a) Contém as configurações necessárias para executar uma aplicação.

b) Contém as informações das classes de cada *Activity.*

c) Possui como elemento raiz a *tag* <manifest></manifest>.

d) Para a aplicação ser iniciada pelo usuário, pelo menos uma *Activity* deve ser configurada como ponto de partida da aplicação.

e) Usamos a tag <intent> com a ação "android.intent.action.MAIN" para identificar uma *Activity* como ponto de partida da aplicação.

45. A classe "android.util.Log" serve para imprimir informações de *log* ao usuário. Qual dos métodos dessa classe imprime uma mensagem, na cor laranja, de alerta ao usuário?

a) Log.d("categoria","mensagem");

b) Log.v("categoria","mensagem");

c) Log.i("categoria","mensagem");

d) Log.w("categoria","mensagem");

e) Log.e("categoria","mensagem");

46. No desenvolvimento Android, o ciclo de vida *foreground lifetime* de uma *Activity* inicia-se e termina na chamada de que métodos, respectivamente?

a) onCreate, onDestroy

b) onStart, onPause

c) onStart, onStop

d) onResume, onPause

e) onResume, onStop

47. Considere as seguintes sentenças sobre o desenvolvimento de uma aplicação Android em Java.

I. A classe “android.app.Notification” deve ser usada para exibir notificações ao usuário.
II. Através da classe “android.content.ContentProvider” criamos provedores de conteúdo para que determinadas informações sejam privadas à aplicação que os criou.
III. Uma das formas de criar um serviço que seja executado em segundo plano é através da definição de uma classe que estenda “android.app.Service” e sobrescreva os métodos “initService” e “bindService”.

Sobre as sentenças anteriores, está (ão) CORRETAS(S):

a) Apenas a sentença I.

b) Apenas a sentença II.

c) Apenas a sentença III.

d) Apenas as sentenças II e III.

e) Apenas as sentenças I e II.

48. Sobre MIDlets, o que se pode considerar FALSO?

a) Uma mensagem SMS que chega a uma determinada porta de conexão pode iniciar um MIDlet.

b) MIDlets podem ser iniciados, automaticamente, em um horário específico .

c) Para que se obtenha um MIDlet, uma implementação da interface javax.microedition.midlet.MIDlet deverá ser realizada.

d) Os métodos startApp, pauseApp e destroyApp pertencem ao ciclo de vida de um MIDlet.

e) Um MIDlet pode ser ativado para manipular arquivos de áudio, automaticamente, sem a interferência do usuário.

49. Dentre as alternativas abaixo, qual delas NÃO oferece vantagens do *E-Commerce* às empresas ou aos consumidores em todo o mundo?

a) Com o E-Commerce, consumidores possuem um comércio global a qualquer hora e a qualquer lugar.

b) Utilizando mecanismos de busca ou agentes inteligentes, consumidores podem facilmente comparar e avaliar produtos.

c) A competição entre empresas agora será vista em uma escala global, e não apenas local como anteriormente.

d) As empresas podem, também, aumentar seu faturamento explorando novas oportunidades, expandindo seus negócios em um comércio global.

e) Uma loja local ou no exterior estão ambas a alguns “clicks” no ciberespaço.

50. No *E-Commerce,* as formas de pagamento (*Cash, Check, Credit Card, Credit/Debit*) podem ser analisadas segundo algumas características. Qual das alternativas abaixo NÃO corresponde a uma dessas características?

a) *Overhead cost*: refere-se ao custo do *overhead* para processar um pagamento.

b) *Security*: preocupa-se com a segurança da forma de pagamento, isto é, se uma forma é ou não fácil de ser fraudada.

c) *Transferability*: se uma forma de pagamento pode ser realizada sem o envolvimento de terceiros, a exemplo um banco.

d) *Commission*: se a forma de pagamento provê algum percentual do valor da compra a uma empresa concessionária.

e) *Acceptability:* verifica se uma forma de pagamento é suportada globalmente, isto é, não estará disponível apenas para um grupo de clientes.