CONHECIMENTOS ESPECÍFICOS
» PROGRAMAÇÃO (PERFIL 1) «

21. Considere uma lista encadeada, cujos nós (contendo dados e elos entre si) estão armazenados nos dois *arrays* abaixo:

	0	1	2	3	4
dados	10	90	30	40	20

	0	1	2	3	4
elos	4	0	3	1	2

Considere também o seguinte trecho de programa, implementado na linguagem C:

```
int ini, i;
scanf("%d",&ini);
i=ini;
do{
  i = elos[i];
}while(i != ini);
printf("%d ",dados[i]);
```

Executando-se esse trecho de programa para o dado de entrada igual a 2, o valor exibido será:

a) 10
b) 90
c) 30
d) 40
e) 20

22. Árvores Binárias de Pesquisa (ABP) são muito utilizadas para a construção de tabelas de símbolos ou de nomes, permitindo que estes possam ser localizados rapidamente pela ordem alfabética. O processo de inserção de um nome, em uma ABP, se desenvolve da seguinte forma:

I. Se a árvore estiver vazia, o nome é inserido na sua raiz.

II. Caso contrário, o nome é inserido na subárvore da esquerda, se for menor que o nome que está na raiz, ou é inserido na subárvore da direita, se for maior que o nome que está na raiz, considerando-se a comparação por ordem alfabética.

Utilizando esse processo, a ABP resultante, a partir da sequência de nomes {joão, paulo, maria, ana, rui}, é:

a)

b)

c)

d)

e)

23. Execute o programa a seguir, implementado na linguagem C:

```
#include <stdio.h>
int v[6]= {5,2,1,4,3,0};
int F (int a,int b,int x) {
        if(a>b) return -1;
        int c=(a+b)/2;
        if (v[v[c]]==x)
           return c;
        else  if (v[v[c]]>x)
                 return F(a,c-1,x);
              else
                 return F(c+1,b,x);
}
int main(void) {
        printf("%d", F(0, 5, 5) );
        return 0;
}
```

Com base na execução desse programa, o número exibido será:

a) -1 b) 0 c) 1 d) 3 e) 5

24. A função utilizada para efetuar a liberação dinâmica de memória de ponteiros, na linguagem C, é identificada por:

a) free

b) dispose

c) finalize

d) disalloc

e) destroy

25. Considere as seguintes classes, implementadas em Java (JSE6):

```
public class Pessoa {
        private String nome;
        private Pessoa gostaDe;
        public Pessoa(String nome)  {this.nome=nome; }
        public Pessoa getAmigo()  { return gostaDe; }
        public void setAmigo(Pessoa p)  { gostaDe=p; }
        public String getNome() {return nome;}
        public String toString() {
        return nome+" gosta de "+((gostaDe==null)? "ninguem": gostaDe.getNome() );
        }
}
public class Teste{
        public static void main (String[] args){
        Pessoa p1, p2, p3;
        p1 = new Pessoa("Maria");
        p2 = new Pessoa("João");
        p3 = new Pessoa("Julia");
        p3.setAmigo(p2);
        p1.setAmigo(null);
        p2.setAmigo(p2);
        p1.setAmigo(p2.getAmigo());
        p3.getAmigo().setAmigo(p1);
        System.out.println(p1);
        System.out.println(p2);
        System.out.println(p3);
}}
```

Analise as mensagens abaixo:

I. Maria gosta de ninguem

II. João gosta de João

III. Julia gosta de João

Dentre elas, qual(is) será(ão) exibida(s) na execução do método **main?**

a) apenas I. b) apenas II. c) apenas III. d) I e II. e) II e III.

ATENÇÃO: Para responder às questões **26** e **27**, considere a interface e as classes implementadas na linguagem Java (JSE6):

```
public interface M {
        public void mostra(int n);
}
public class A implements M {
        public void mostra(int n)     { System.out.println(n+1);}
        public void mostra ()         { mostra(3); }
}
public class B extends A {
        public void mostra (int n)    { super.mostra (n+1); }
        public void mostra ()         { super.mostra (); }
}
```

26. Analise o trecho de programa abaixo:

```
B b1 =new B();
b1.mostra(2);
b1.mostra();
```

Quais os números exibidos pela sua execução?

a) 3 e 4

b) 3 e 5

c) 4 e 4

d) 4 e 5

e) 5 e 5

27. Considerando, além das classes apresentadas, a existência de outra classe Java com métodos identificados pelos nomes **p, q** e **r**, assinale (V) para Verdadeiro ou (F) para Falso, para as seguintes afirmações:

( ) O método **void p(M m1***)* pode receber, como parâmetro, um objeto da classe **A**

( ) O método **void q(A a1)** pode receber, como parâmetro, um objeto da classe **B**

( ) O método **void r(A a1)** pode ser sobrecarregado (*overloaded*) pelo método **void r(B b1)**

( ) A implementação do método **void p(M m1){m1.mostra();}** compila sem erro

A sequência CORRETA das respostas assinaladas é:

a) V, V, V, V

b) V, F, F, V

c) F, V, F, V

d) F, V, V, F

e) V, V, V, F

28. Considere os métodos abaixo, pertencentes a uma classe Java (JavaSE6):

```
public void tarefa(){
        try{System.out.print("1");  passo1();   System.out.print("2");}
        catch (Exception e) {System.out.print(e.getMessage());}
 }
public void passo1() throws Exception {
        try{ System.out.print("3");  passo2();  System.out.print("4");          }
        catch(Exception e) {System.out.print(e.getMessage()); throw new Exception("5"); }
 }
public void passo2() throws Exception {
        System.out.print("6");  throw new Exception("7");
 }
```

Executando-se o método **tarefa**, qual a sequência correta de valores que serão exibidos pela sua execução?

a) 13642 b) 136427 c) 136475 d) 13675 e) 13677

29. Preencha as lacunas dos trechos de código fonte Java (JSE6) abaixo, considerando a variável **vendas**, definida como **ArrayList<Venda>**:

I. for (_____ v : vendas) {if(v.getCod()==2) return v.getData(); }

II. public void a() {vendas = new _____ ();}

III. public Venda b(_____ i) {return vendas.get(i);}

IV. public void c() {vendas.add(new _____ ());}

A sequência CORRETA de preenchimento das lacunas é:

a) Venda/ArrayList<Venda>/int/Venda

b) ArrayList/ArrayList<Venda>/String/Venda

c) Venda/Venda,/int/ ArrayList<Venda>

d) ArrayList/Venda/int/Venda

e) Venda/ArrayList<Venda>/String/ArrayList<Venda>

30. Com respeito ao ciclo de vida de objetos em Java (JSE6), observe o trecho de código fonte abaixo:

```
Object r1,r2,r3;
r1=new Object();
r2=new Object();
r3=r2;
r1=new Object();
r2=r1;
r1=null;
```

Após a execução desse trecho, quantos objetos estarão elegíveis para a coleta de lixo de memória?

a) 0 b) 1 c) 2 d) 3 e) 4

ATENÇÃO: Para responder as questões **31** e **32**, considere a seguinte situação:

Em um *forum* eletrônico de discussão, várias mensagens são postadas a partir dos usuários e cada mensagem pode ter vários comentários, também postados por qualquer um dos usuários. As mensagens obedecem a uma sequência única de identificação (id=1, 2, 3,...). Os comentários também são identificados sequencialmente, porém a sua identificação é reiniciada em 1 para cada nova mensagem (por exemplo, pode existir um comentário com id=5 para a mensagem com id=42 e um outro com id=5 para a mensagem com id=127). Essas informações estão representadas pelas seguintes relações:

Usuario (email, senha)
Mensagem(id, emailusuario, texto, data)
Comentario (idmensagem, idcomentario, emailusuario, texto, data)

31. Os atributos que definem, respectivamente, as chaves primárias (sublinhadas) e as chaves estrangeiras (em *itálico*) dessas relações, são:

a) Usuario (email, senha)
Mensagem(id, *emailusuario*, texto, data)
Comentario (idmensagem, idcomentario, *emailusuario*, texto, data)

b) Usuario (email, senha)
Mensagem(id, *emailusuario*, texto, data)
Comentario (*idmensagem,* idcomentario, *emailusuario*, texto, data)

c) Usuario (email, senha)
Mensagem(id, emailusuario, texto, data)
Comentario (*idmensagem,* idcomentario, *emailusuario*, texto, *data*)

d) Usuario (email, senha)
Mensagem(id, emailusuario, texto, data)
Comentario (*idmensagem*, *idcomentario*, emailusuario, texto, data)

e) Usuario (email, senha)
Mensagem(id, *emailusuario*, texto, data)
Comentario (*idmensagem*, idcomentario, *emailusuario*, texto, data)

32. Analise o comando SQL a seguir:

Select idmensagem, count(*) from comentario group by idmensagem having count(*)>2

Pode-se entender esse comando da seguinte forma:

a) Obter a quantidade de comentários das 2 primeiras mensagens.
b) Obter a quantidade de comentários de cada mensagem com mais de 2 comentários.
c) Obter a quantidade de comentários da mensagem que tem a maior quantidade de comentários.
d) Obter a quantidade de mensagens com mais de 2 comentários.
e) Obter as 2 primeiras mensagens com mais de 2 comentários.

ATENÇÃO: Para responder às questões **33** e **34**, considere o cenário de uma empresa, contendo departamentos, projetos e empregados, e as relações abaixo representadas, em que os atributos sublinhados são chaves primárias e os atributos em *itálico* são as chaves estrangeiras.

Departamento (Dn, Dnome, *Chefe*)
Empregado (Matric, Enome, Salario, *Dn*)
Projeto (Pn, Pnome, Duracao, *Dn*, *Chefe*)

Sobre essas relações, pode-se dizer:

I. Em cada departamento, trabalham vários empregados e desenvolvem-se vários projetos.

II. Cada departamento e cada projeto têm um empregado chefe.

III. Cada empregado trabalha em apenas um departamento, mas pode chefiar zero, um, ou mais departamentos e/ou chefiar zero, um ou mais projetos.

33. Qual o comando SQL que obtem as matrículas dos chefes de projeto que não chefiam projetos do departamento D1?

a) select distinct e.matric from empregado e
where not exists (select * from projeto p2 where p2.chefe=e.matric and p2.dn='D1')

b) select distinct e.matric from empregado e, projeto p1
where e.matric=p1.chefe and p1.dn='D1'

c) select distinct p1.chefe from projeto p1
where not exists (select * from projeto p2 where p1.chefe=p2.chefe and p2.dn='D1')

d) select distinct p1.chefe from projeto p1
where not exists (select * from projeto p2 where p1.chefe=p2.chefe and p1.dn='D1')

e) select distinct e.matric from empregado e
where exists (select * from projeto p2 where p2.chefe=e.matric and p2.dn<>'D1')

34. Analise o comando SQL a seguir:

```
Select e1.matric
From empregado e1, departamento d
Where  e1.matric=d.chefe and
        e1.salario < (select avg(salario) from empregado e2  where d.dn=e2.dn)
```

Pode-se entender esse comando da seguinte forma:

a) Obter os chefes de departamento que ganham menos do que a média salarial de todos os outros empregados da empresa.

b) Obter os chefes de departamento que ganham menos do que a média salarial de todos os outros empregados que trabalham no mesmo departamento em que esses chefes chefiam.

c) Obter os chefes de departamento que ganham menos do que a média salarial de todos os outros empregados que trabalham nos departamentos em que esses chefes não chefiam.

d) Obter os chefes de departamento que ganham menos do que a média salarial de todos os outros empregados que trabalham no mesmo departamento em que esses chefes trabalham.

e) Obter os chefes de departamento que ganham menos do que a média salarial de todos os outros empregados que trabalham nos departamentos em que esses chefes não trabalham.

35. Sobre transação em um Banco de Dados Relacional, considere as seguintes assertivas:

I. As propriedades ACID de uma transação significam, respectivamente: Atomicidade, Complexidade, Isolamento e Deletabilidade.

II. O nível de isolamento *read uncommited* (leitura não efetivada) é o mais seguro, pois impede que uma transação realize leitura de dados sujos ou de dados fantasmas.

III. Quando ocorre um *rollback* (cancelamento), a transação é interrompida, e, neste caso, todas as operações de gravação de dados realizadas dentro da transação, até a sua interrupção, são efetivadas no banco de dados.

Qual(is) assertiva(s) está(ão) INCORRETA(S)?

a) apenas III.

b) I e II.

c) I e III.

d) II e III.

e) I, II e III.

36. Na especificação de Servlets 3.0, uma classe que implementa a interface **ServletRequestListener** deve implementar os seguintes métodos:

a) requestInit e requestDestroy

b) requestInitialized e requestDestroyed

c) requestCreated e requestDestroyed

d) createdRequest e destroyedRequest

e) initializedRequest e destroyedRequest

37. Sobre atributos, de acordo com a especificação de Servlets 3.0, é INCORRETO afirmar:

a) Atributos só podem ser encapsulados em objetos que implementam as interfaces ServletRequest, HttpSession e ServletContext.

b) O método getAttribute possui o tipo de retorno Object.

c) Atributos podem ser definidos no arquivo web.xml.

d) Um atributo pode ser um objeto de qualquer classe.

e) Atributos podem ser passados entre requisições, quando é realizado um despacho de requisição, usando um objeto RequestDispatcher e o método forward.

38. Sobre o ciclo de vida do JSF (*Java Server Faces*), as fases ocorrem na seguinte ordem, respectivamente:

a) Restore View, Apply Request Values, Invoke Application, Process Validations, Update Model Values e Render Response.

b) Render Response, Update Model Values, Process Validations, Apply Request Values, Invoke Application e Restore View.

c) Render Response, Apply Request Values, Process Validations, Update Model Values, Invoke Application e Restore View.

d) Restore View, Apply Request Values, Process Validations, Update Model Values, Invoke Application e Render Response.

e) Restore View, Update Model Values, Process Validations, Apply Request Values, Invoke Application e Render Response.

39. Considere o *Backing Bean* **bb** com as propriedades **valor1** e **valor2** do tipo String e inicializadas com "8" e "alo", respectivamente. Dados os seguintes trechos de código fonte, analise as preposições que seguem:

C1:

```
<h:outputFormat value="{0},{1}">
<f:param value="#{bb.valor2}" />
<f:param value="#{bb.valor1}" />
</h:outputFormat>
```

C2:

```
<h:outputText value="#{bb.valor1 + 2}" />
```

C3:

```
<h:outputLink value="listarItens">
<h:outputText value="Codigo" />
<f:param name="cod" value="#{bb.valor1}" />
</h:outputLink>
```

I. O código C1 tem como saída impressa no *browser* a mensagem: alo,8

II. O código C2 tem como saída impressa no *browser* a mensagem: 82

III. O código C3 tem como saída impressa no *browser* um *link* para a página listarItens, que contém o parâmetro de nome cod com valor 8

É CORRETO o que se afirma em:

a) I apenas. b) II apenas. c) III apenas. d) I e III. e) I e II.

40. Uma classe Java chamada **Cliente**, possui a propriedade **endereco**. O valor dessa propriedade é um objeto do tipo **Endereco** com as seguintes propriedades: **rua, cidade, estado** e **cep**. Um servlet controlador cria um atributo de sessão chamado **cliente**, que é uma instância da classe **Cliente**. Considerando essas especificações, que código JSP (*Java Server Pages*) irá alterar a propriedade **cidade** de **cliente** pelo valor do parâmetro de requisição **cidade**?

a) ${sessionScope.cliente.endereco.cidade = param.cidade}

b) <c:set target="${sessionScope.cliente.endereco}" property="cidade" value="${param.cidade}" />

c) <c:set scope="session" var="${cliente.endereco}" property="cidade" value="${param.cidade}" />

d) <c:set target="${requestScope.cliente.endereco}" property="cidade" value="${param.cidade}" />

e) <c:set scope="request" var="${cliente.endereco}" property="cidade" value="${param.cidade}" />

41. Sobre JPA (*Java Persistence API*), analise as preposições a seguir:

I. A anotação **@javax.persistence.GeneratedValue** possui, dentre suas possíveis estratégias, os valores: GenerationType.IDENTITY e GenerationType.SEQUENCE.

II. A anotação **@javax.persistence.Transient** serve para identificar que uma propriedade, da entidade, não será persistida no banco de dados.

III. A anotação **@javax.persistence.Lob** serve para identificar que uma propriedade da entidade deve ser persistida no banco de dados em um campo do tipo *Blob* ou do tipo *Clob.*

É CORRETO o que se afirma em:

a) I apenas.

b) III apenas.

c) I e II.

d) II e III.

e) I, II e III.

42. O gerenciador de entidades (EntityManager*)* da *Java Persistence API*, persiste um objeto no banco de dados, usando o método **persist,** e, logo em seguida, carrega um outro objeto, usando o método **find**. Que eventos são gerados, em sequência, envolvendo essas duas operações?

a) @PrePersist, @PostPersist, @PreLoad, @PostLoad

b) @PostPersist,@PostLoad

c) @PostPersist, @PreLoad, @PostLoad

d) @PrePersist, @PostPersist, @PostLoad

e) @PrePersist, @PreLoad

43. Sobre Beans de Sessão Singleton, analise as afirmações que seguem:

I. São classes anotadas com @javax.ejb.Singleton e só existe uma única instância controlada pelo *container*

II. @javax.ejb.Startup serve para forçar o *container* a instanciar o Singleton na inicialização da aplicação

III. @javax.ejb.Lock(Write) e @javax.ejb.Lock(Read) servem para controlar o acesso concorrente aos métodos do Singleton. Por *default* todos os métodos são anotados com @javax.ejb.Lock(Read)

É CORRETO o que se afirma em:

a) I e II.

b) I, II e III.

c) II e III apenas.

d) I apenas.

e) III apenas.

44. Sobre Beans Baseados em Mensagens, é INCORRETO afirmar:

a) *O Java Message Service* (JMS) fornece os modelos de sistema de mensagens: publicação e assinatura e ponto a ponto.

b) Um Bean Baseado em Mensagem processa mensagens assíncronas que são recebidas pelo container.

c) Um Bean Baseado em Mensagem deve ser uma classe anotada com @javax.ejb.MessageDriven, com o método chamado message, responsável por processar a mensagem.

d) Os Beans Baseados em Mensagem são instanciados aleatoriamente pelo container e não possuem informações de estado conversacional.

e) As anotações @javax.ejb.PostConstruct e @javax.ejb.PreDestroy podem ser usadas em métodos dos Beans Baseados em Mensagens.

45. Os servidores EJB (*Enterprise Java Beans*) podem gerenciar transações, implicitamente, com base nos atributos de transação, estabelecidos em tempo de implantação. Um desses atributos significa que o método do *enterprise bean*, que possui tal atributo, deve ser invocado dentro do escopo de uma transação. Dessa forma, se o cliente ou EJB chamador fizerem parte de uma transação, o método do EJB que usa esse atributo é automaticamente incluído ao escopo do cliente ou EJB chamador. Se o cliente ou EJB chamador não fizerem parte de uma transação, o método do EJB, que usa esse atributo, inicia sua própria nova transação. Considerando essas informações, esse atributo é:

a) SUPPORTS

b) REQUIRED

c) REQUIRES_NEW

d) MANDATORY

e) NEVER

46. Sobre a criação de *Web Services*, utilizando a API JAX-WS, é INCORRETO afirmar:

a) A anotação @javax.jws.WebService é utilizada para identificar uma classe Java como sendo um *WebService.*

b) Os métodos anotados com @javax.jws.WebMethod ficam disponíveis ao serviço. Se nenhum método for anotado com essa anotação, nenhum estará disponível.

c) O *container* gera automaticamente o WSDL da classe anotada com @javax.jws.WebService que foi implantada.

d) A anotação @javax.jws.OneWay é utilizada para declarar que a operação do *Web Service* retornará uma mensagem vazia.

e) A anotação @javax.xml.ws.WebServiceRef é utilizada para injetar uma referência a uma classe ou interface de serviço.

47. Considere uma aplicação Android que possui duas telas: Tela1 e Tela2. A Tela2 é executada a partir da Tela1, e, logo depois, é pressionado o botão voltar (botão padrão do Android). Diante desse cenário, quais métodos do ciclo de vida da Tela1 são executados, em sequência, logo depois de pressionado o botão voltar?

a) onRestart, onStart, onResume

b) onRestart, onResume

c) onCreate, onResume

d) onPause, onStart, onResume

e) onStart, onRestart, onResume

48. Objetos da classe **android.content.Intent** representam uma ação que a aplicação deseja executar. Como base nessa afirmação, dentre os exemplos a seguir, qual NÃO representa a utilização de um objeto desta classe?

a) Enviar uma mensagem para o Sistema Operacional.

b) Abrir uma nova tela da aplicação, utilizando o método openActivity.

c) Solicitar ao Sistema Operacional que ligue para algum número de celular.

d) Exibir algum endereço no Google Maps.

e) Abrir algum endereço de internet no Browser.

49. Em relação ao desenvolvimento de aplicações em Android, é CORRETO afirmar:

a) A classe android.view.View é superclasse de classes que definem elementos visuais.

b) A classe R.java contém informações de configuração necessárias a execução da aplicação.

c) O método setContent, da classe android.app.Activity, faz a ligação entre uma *Activity* e uma *View.*

d) O ciclo de vida *visible lifetime* de uma *Activity*, inicia-se e termina na chamada dos métodos onResume e onPause, respectivamente.

e) A tag <intent> é usada para configurar uma *Intent* (android.content.Intent) no arquivo AndroidManifest.xml.

50. Sobre o ciclo de vida de um MIDlet, é INCORRETO afirmar:

a) Possui três estados possíveis: Active, Paused e Destroyed.

b) Os métodos que transitam entre estados são: startApp, pauseApp e destroyApp.

c) O método startApp será chamado apenas uma vez durante o ciclo de vida de um MIDlet.

d) O método pauseApp pode ser chamado várias vezes durante o ciclo de vida de um MIDlet.

e) O método destroyApp recebe, como parâmetro, uma variável do tipo boolean.