SERVIÇO PÚBLICO FEDERAL
MEC / SETEC

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE

Departamento de Seleção

CAMPUS PASSO FUNDO

INSTRUÇÕES GERAIS

1 - Este caderno de prova é constituído por 40 (quarenta) questões objetivas.

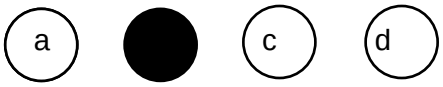
2 - A prova terá duração máxima de 04 (quatro) horas.

3 - Para cada questão são apresentadas 04 (quatro) alternativas (a – b – c – d). **APENAS *UMA* delas** constitui a resposta CORRETA.

4 - Após conferir os dados contidos no campo "Identificação do Candidato" no Cartão de Resposta, assine no espaço indicado.

5 - As alternativas assinaladas deverão ser transcritas para o Cartão de Resposta, que é o único documento válido para correção eletrônica.

6 - Marque o Cartão de Resposta conforme o exemplo abaixo, com caneta esferográfica azul ou preta, de ponta grossa:



7 - **Em hipótese alguma haverá substituição do Cartão de Resposta.**

8 - Não deixe nenhuma questão sem resposta.

9 - O preenchimento do Cartão de Resposta deverá ser feito dentro do tempo previsto para esta prova, ou seja, 04 (quatro) horas.

10 - Serão anuladas as questões que tiverem mais de uma alternativa marcada, emendas e/ou rasuras.

11 - O candidato só poderá retirar-se da sala de prova após transcorrida 01 (uma) hora do seu início.

12 - Não é permitido o uso de calculadora.

BOA PROVA!

Edital 068/2011 - Área 23 – Informação e Comunicação

01. A respeito de interfaces em Java, é correto afirmar que

a) uma classe concreta que implementa uma interface deve declarar todos os métodos da interface, caso contrário, um erro de compilação ocorrerá.
b) uma classe abstrata que implementa uma interface deve declarar todos os métodos da interface, caso contrário, um erro de compilação ocorrerá.
c) a palavra-chave *extends* é utilizada para especificar que uma classe implementa uma interface.
d) a palavra-chave *matcher* é utilizada para especificar que uma classe implementa uma interface.

02. Em se tratando do paradigma de orientação a objetos, analise as seguintes afirmações sobre o conceito de herança:

I. Uma subclasse é uma especialização de uma superclasse.
II. Uma superclasse é uma classe que é estendida por outra classe.
III. No máximo duas subclasses podem herdar da mesma superclasse.

Estão corretas as afirmativas

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

03. Analise as seguintes afirmações sobre herança em Java:

I. Uma subclasse não herda os atributos e métodos privados de sua superclasse.
II. O construtor da subclasse sempre invoca o construtor da sua superclasse.
III. Uma subclasse que herda de duas ou mais superclasses estabelece um relacionamento de herança múltipla.

Está(ao) correta(s) apenas a(s) afirmativa(s)

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

04. Uma classe declarada com o modificador *final* **NÃO** pode

a) ser uma superclasse.
b) ser uma subclasse.
c) ter métodos estáticos.
d) ser instanciada.

05. A respeito dos métodos de classe ou métodos estáticos da linguagem Java, é **INCORRETO** afirmar que.

a) Um método estático definido na classe A pode ser invocado sem a necessidade de criar uma instância da classe A.
b) Um método estático definido na classe A não pode acessar diretamente uma variável de instância da classe A.
c) Um método estático definido na classe A não pode invocar diretamente um método de instância da classe A.
d) Um método de instância da classe A não pode invocar diretamente um método estático da classe A.

Considere o código-fonte Java abaixo para responder às questões 6, 7 e 8.

```java
package provaprof2011;

public abstract class Pessoa {
    private String nome;
    private int rg;
    private int fone;

    public Pessoa(){
        nome = "Fulano Silva";
        rg = 123456789;
        fone = 30451234;
    }

    public void imprimir(){
        String msg = "";
        msg += "\nNome:"+nome;
        msg += "\nRG:"+rg;
        msg += "\nFone:"+fone;
        System.out.println(msg);
    }
}
```

```java
package provaprof2011;

public class Professor extends Pessoa {
    private int horas;
    private double salario;

    public Professor(){
        horas = 20;
        salario = 1000.0;
    }

    public void imprimir(){
        String msg = "";
        msg += "\nHoras:"+horas;
        msg += "\nSalário:"+salario;
        System.out.println(msg);
    }
}
```

```java
package provaprof2011;

public class Aluno extends Pessoa {
    // Matrícula do aluno
    private String matri;

    public Aluno(){
        matri = "123-45";
    }

    public void imprimir(){
        String msg = "";
        msg += "\nMatrícula:"+matri;
        System.out.println(msg);
    }
}
```

06. O método *imprimir()* nas classes Professor e Aluno é um exemplo de

a) sobrecarga de método.
b) sobrescrita de método.
c) extensão de método.
d) definição de métodos abstratos.

07. Qual das seguintes instruções gera erro de compilação?

a) Pessoa p1 = new Pessoa();
b) Pessoa p1 = new Aluno();
c) Professor p1 = new Professor();
d) Pessoa p1 = new Professor();

08. Analise o código-fonte abaixo.

```
1  package provaprof2011;
2
3  public class Teste {
4    public static void main(String[] args){
5      Aluno a1 = new Aluno();
6      a1.imprimir();
7    }
8  }
```

A saída gerada pelo programa acima imprime no console o(s) dado(s) do(s) atributo(s)

a) nome, RG, fone, matrícula, horas e salário do objeto a1.
b) nome, RG, fone e matrícula do objeto a1.
c) nome, RG e fone do objeto a1.
d) matrícula do objeto a1.

09. O nome da interface JDBC (*Java Database Connectivity*) que permite criar instruções SQL compiladas e parametrizadas é

a) Connection.
b) Statement.
c) PreparedStatement.
d) DriverManager.

10. O nome da interface JDBC (*Java Database Connectivity*) utilizada para manipular o conjunto de registros retornado por uma consulta SQL ao banco de dados é

a) Query.
b) RecordSet.
c) ResultSet.
d) CallableStatement.

11. Analise o trecho de código abaixo.

```
1  package provaprof2011;
2
3  public class Lampada {
4    private String estado;
5
6    public void ligar(){
7      estado = "ligado";
8    }
9
10   public void desligar(){
11     estado = "desligado";
12   }
13
14   public void print(){
15     System.out.print(estado+"; ");
16   }
17 }
```

```
1  package provaprof2011;
2
3  public class UsaLampada {
4    public static void main(String[] args){
5      Lampada lamps[] = new Lampada[3];
6      lamps[0] = new Lampada();
7      lamps[1] = new Lampada();
8      lamps[2] = new Lampada();
9
10     int i = 0;
11     while (i<10){
12         int indice = i % lamps.length;
13         if ((i % 2)==0)
14             lamps[indice].ligar();
15         else
16             lamps[indice].desligar();
17         i++;
18     }
19
20     lamps[0].print();
21     lamps[1].print();
22     lamps[2].print();
23   }
24 }
```

A saída gerada pelo programa acima é

a) ligado; ligado; desligado;
b) ligado; desligado; ligado;
c) desligado; ligado; ligado;
d) desligado; desligado; ligado;

12. Em uma aplicação JSP, o comando para redirecionar o navegador do usuário para a página index.jsp é

a) response.sendRedirect(“index.jsp”);
b) request.sendRedirect(“index.jsp”);
c) response.location(“index.jsp”);
d) request.location(“index.jsp”);

13. Para aumentar a produtividade do programador, estão disponíveis alguns objetos que podem ser usados dentro de uma página JSP sem necessidade de instanciação. Esses objetos são conhecidos como objetos implícitos do JSP.

Que alternativa **NÃO** apresenta um objeto implícito do JSP?

a) session.
b) cookie.
c) out.
d) application.

14. Analise o código-fonte dos arquivos *login.jsp* e *exibirDados.jsp* abaixo:

```
1  <%-- login.jsp --%>
2  <html><body>
3    <form action="exibirDados.jsp" >
4      Login:<input type="text" name="login"/><br>
5      Senha:<input type="password" name="senha"/><br>
6      <input type="submit" value="Ok"/>
7    </form>
8  </body></html>
```

```
1  <%-- exibirDados.jsp --%>
2  <html><body>
3    Login = <%=_____%>
4    <br>
5    Senha = <%=_____%>
6  </body></html>
```

O arquivo *exibirDados.jsp* está incompleto. A alternativa que apresenta os comandos corretos para preencher as lacunas das linhas 3 e 5, respectivamente é

a) request.getParameter("login") e request.getParameter("senha")
b) request.getAttribute("login") e request.getAttribute("senha")
c) response.getParameter("login") e response.getParameter("senha")
d) response.getAttribute("login") e response.getAttribute("senha")

15. A respeito dos modificadores de acesso do Java, é correto afirmar que membros *protected* de uma

a) classe são acessíveis onde quer que o programa tenha uma referência a um objeto dessa classe.
b) superclasse podem ser acessados pela própria superclasse e também pelas suas subclasses, desde que as classes estejam localizadas no mesmo pacote.
c) superclasse podem ser acessados pela própria superclasse e também pelas suas subclasses, independente do pacote onde as classes estejam localizadas.
d) superclasse não são herdados pelas suas subclasses.

16. Analise o trecho de código abaixo.

```
1  package provaprof2011;
2
3  public class Carro {
4      private String placa;
5      private int renavam;
6      private String cor;
7      // Primeiro construtor
8      public Carro(String p, int r){
9          placa = p;
10         renavam = r;
11     }
12     // Segundo construtor
13     public Carro(String c, int r){
14         cor = c;
15         renavam = r;
16     }
17     // Terceiro construtor
18     public Carro(int r, String c){
19         cor = c;
20         renavam = r;
21     }
22 }
```

Em relação ao trecho de código, é correto afirmar que.

a) A classe Carro contém erro(s). O primeiro construtor e o segundo construtor tem a mesma assinatura.
b) A classe Carro contém erro(s). O segundo construtor e o terceiro construtor inicializam os mesmos campos da classe.
c) A classe Carro contém erro(s). Nenhum construtor inicializa todos os campos da classe.
d) A classe Carro não contém erros e será compilada com sucesso.

17. Dado o trecho de código Android a seguir:

```
 1 public class TesteActivity extends Activity {
 2
 3     // trecho de código omitido
 4     public void onClick(View v) {
 5         Intent it = new Intent(this,Tela2.class);
 6         it.putExtra("msg","Ola mundo");
 7         startActivity(it);
 8     }
 9     // trecho de código omitido
10 }
```

Analise as seguintes afirmações:

I. O objeto *it*, instância da classe *Intent*, identifica a *Activity* que será exibida.
II. O método *startActivity(it)* informa ao sistema operacional que a aplicação quer abrir uma nova tela.
III. O método *putExtra("msg","Ola Mundo")*, do objeto *it*, é responsável pelo envio de um parâmetro para a *activity* chamada.

Estão corretas as afirmativas

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

18. Existem diversas plataformas do *Android* circulando no mercado, todas identificadas por um código chamado *API Level*. A alternativa que corresponde à *API Level* da plataforma *Android 2.2* é

a) API Level 6
b) API Level 7
c) API Level 8
d) API Level 9

19. Um objeto *Activity*, especificado pela classe *android.app.Activity*, é peça fundamental da plataforma *Android* e representa basicamente

a) a troca de informações entre aplicações.
b) o contexto de execução de uma aplicação.
c) uma tela de uma aplicação.
d) um widget de uma tela.

20. Ambientes baseados em computação móvel e/ou ubíqua são considerados sistemas voláteis. Sobre essa questão, apresentam-se as seguintes causas:

I. Porque o conjunto de usuários em um ambiente de computação móvel é altamente dinâmico.
II. Porque os dispositivos podem entrar e sair em um ambiente de computação móvel com facilidade.
III. Porque os serviços disponíveis em um ambiente de computação móvel não são modificados, ou seja, permanecem ativos independentemente dos outros componentes.

Estão corretas as afirmativas

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

21. Em se tratando de computação móvel, afirmam-se que:

I. Confiança é um atributo comum em redes móveis e/ou ubíquas.
II. Frequentemente, em redes de dispositivos móveis, ocorre interação espontânea entre dispositivos.
III. Ambientes inteligentes são a base da computação ubíqua já que os dispositivos móveis estão incorporados ao meio.
IV. Um espaço inteligente deve, obrigatoriamente, possuir uma infraestrutura prévia para associação de novos dispositivos.

Estão corretas as afirmativas

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

22. Sobre percepção e reconhecimento de contexto, dentro de computação móvel, é **INCORRETO** afirmar que

a) sensores são utilizados para coletar dados do ambiente.
b) o contexto de uma entidade é um conjunto de circunstâncias físicas de relevância para o comportamento de determinado sistema.
c) sensores podem coletar diversos parâmetros, como, por exemplo, velocidade, localização, orientação e presença.
d) sensores podem se intercomunicar exclusivamente através de redes cabeadas.

23. Sobre computação móvel, no que diz respeito à arquitetura de sensores, afirmam-se que:

I. Sensores idiossincráticos são aqueles considerados incomuns ou instalados em locais incomuns.
II. Na leitura dos sensores, é necessária uma abstração adequada para evitar erros de interpretação dos dados.
III. Sensores com informações diferentes não podem ser combinados.

Estão corretas as afirmativas

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

24. É possível aplicar uma regra CSS comum a vários seletores utilizando o recurso de agrupamento. A forma correta de agrupar seletores é

a) h1 h2 p {color: red}
b) h1; h2; p {color: red}
c) h1, h2, p {color: red}
d) (h1 h2 p) {color: red}

25. Analise a declaração CSS a seguir: **margin: 20px 30px 5px 10px;** A declaração acima define o tamanho das quatro margens de um *box* CSS, atribuindo os valores na seguinte ordem:

a) superior, inferior, direita e esquerda.
b) direita, esquerda, superior e inferior.
c) superior, direita, inferior e esquerda.
d) esquerda, superior, direta e inferior.

26. Analise o trecho de código HTML e a regra CSS a seguir:

```
1 <body>
2     <div>
3         <p>Um parágrafo.</p>
4         <p>Mais um parágrafo.</p>
5         <h2>Cabeçalho de nível 2.</h2>
6     </div>
7 </body>
```

```
1
2 div p:first-child {font-size: 20px}
3
```

Que parte do documento HTML é formatado pela regra CSS apresentada acima?

a) A linha 3 é alvo da formatação.
b) A linha 4 é alvo da formatação.
c) A linha 5 é alvo da formatação.
d) As linhas 3, 4 e 5 são alvos da formatação.

27. Analise o trecho HTML a seguir:

```
1 <html><body>
2     <form name="form1">
3         <input type="submit" value="Ok">
4     </form>
5 </body></html>
```

O JavaScript permite referenciar os formulários de um documento HTML via objeto *document*. A alternativa **INCORRETA** em relação à forma de referenciar o formulário *form1* é

a) document.forms[0]
b) document.form1
c) document.forms.form1
d) document."form1"

28. O tratador de evento *onblur* do JavaScript é disparado quando o elemento HTML associado

a) perde o foco.
b) recebe o foco.
c) tem seu valor alterado pelo usuário.
d) recebe um clique do mouse.

29. Analise as afirmações sobre declaração e uso de funções em JavaScript:

I. Uma função que não possui comando *return* em seu corpo retorna o valor *undefined* para quem a chamou.
II. Em JavaScript é possível definir funções aninhadas, ou seja, declarar uma função dentro do corpo de outra função.
III. Passar um número maior de argumentos do que foi declarado em uma função não gera erro de execução. Os valores extras serão simplesmente ignorados.

Estão corretas as afirmativas

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

30. Analise o trecho de código JavaScript:

```
1 <script>
2     function check(a,b){
3         return (a===b);
4     }
5 </script>
```

No código apresentado acima, a função *check*

a) retorna verdadeiro somente se os valores de *a* e *b* forem iguais.
b) retorna verdadeiro se os tipos de dados e os valores de *a* e *b* forem iguais.
c) atribui o valor de *b* para *a* e retorna verdadeiro se os valores de *a* e *b* forem iguais.
d) gera um erro de execução.

31. Sobre serviços web SOAP, quando transportados sobre HTTP, é **INCORRETO** afirmar que

a) a mensagem SOAP é transportada no corpo do protocolo HTTP.
b) os endereços IP do destinatário e do remetente de uma mensagem SOAP devem constar no envelope da mensagem SOAP.
c) os cabeçalhos HTTP especificam a URI do receptor final e a ação a ser executada pelo serviço web.
d) as requisições HTTP GET ou HTTP POST podem ser utilizadas, dependendo da situação.

32. Em se tratando da arquitetura de serviços web SOAP, que alternativa apresenta a finalidade de um arquivo escrito em WSDL (*Web Services Description Language*) na referida arquitetura?

a) Descrever a forma e a linguagem de programação nas quais o serviço foi implementado, descrevendo os *stubs* para interoperabilidade com serviços heterogêneos.
b) Descrever a localização e o protocolo de comunicação do provedor de serviços web, permitindo o acesso transparente dos clientes aos serviços hospedados no servidor.
c) Descrever um serviço web detalhadamente, especificando as operações que compõem o serviço, definindo o formato de entrada e saída de cada operação.
d) Implementar as operações de um serviço web, permitindo o funcionamento concreto do serviço em um provedor de serviços web.

33. Sobre a arquitetura de serviços web SOAP, que alternativa apresenta somente sentenças verdadeiras a respeito do objetivo da UDDI (Universal Description, Discovery and Integration) na referida arquitetura?

a) Fornecer a descrição de *templates* de serviços web, de modo que desenvolvedores possam fazer a descoberta desses *templates* para a criação de serviços web personalizados.
b) Construir URIs para serviços web publicados em um serviço de diretório de terceiros e integra serviços web usando arquivos WSDL como *stubs* de interoperabilidade.
c) Descrever serviços web publicados, atuando como automatizador de serviços de descoberta e integração para clientes de arquiteturas heterogêneas.
d) Fornecer ao provedor de serviços web meios para que os serviços web sejam registrados e publicados, permitindo assim que esses possam ser localizados pelos clientes. Também tem a finalidade de armazenar arquivos WSDL.

34. Em se tratando de serviços web REST, a respeito do controle de estado de solicitações, que alternativa apresenta afirmações corretas?

a) Serviços web REST são propostos para suportar a retomada de estado de uma transação em caso de falha do serviço web.
b) Serviços web REST são propostos para oferecer a possibilidade de, durante o tratamento de uma requisição, efetuar a recuperação de estados de transações anteriores, desde que sejam do mesmo cliente e não tenham ocorrido falhas no serviço web.
c) Serviços web REST não fornecem suporte à recuperação de estado de solicitações anteriores. No entanto, armazenam o estado da solicitação do serviço atual para recuperar o mesmo em caso de falha do serviço web.
d) Serviços web REST não fornecem suporte à recuperação de estado de solicitações, isso significa que toda solicitação HTTP ocorre em um isolamento completo, o servidor nunca conta com as informações das solicitações anteriores.

35. No contexto de serviços web REST, que alternativa apresenta uma funcionalidade do método HTTP HEAD para uso em solicitações de um cliente a um serviço web?

a) Recuperar metadados sobre um recurso sem recuperar sua representação inteira.
b) Enviar, junto à solicitação ao serviço web, algum outro protocolo através de um Proxy HTTP.
c) Verificar quais métodos HTTP um determinado recurso suporta.
d) Iniciar uma autenticação com o serviço, enviando informações de cabeçalho do cliente.

36. Qual o padrão de projeto que é utilizado para fornecer um mecanismo que permita acessar sequencialmente os elementos de um objeto agregador (que agrega outros objetos) sem expor a sua estrutura interna?

a) Proxy.
b) Visitor.
c) Composite.
d) Iterator.

37. A respeito da camada controle da arquitetura MVC (*Model-View-Controller*), que alternativa contém somente afirmações corretas.

a) Fornece uma apresentação da camada modelo para o usuário da aplicação e é responsável pelo armazenamento de dados da aplicação.
b) Recebe instruções da camada de visualização, determina quais as ações a serem efetivadas na camada modelo e pode solicitar mudanças na camada visualização.
c) É responsável pelo armazenamento de dados da aplicação, recebe instruções da camada visualização e determina quais as ações a ser efetivadas na camada modelo.
d) Pode solicitar mudanças na camada visualização e fornece uma apresentação da camada modelo para o usuário da aplicação.

38. Em um projeto de *software* orientado a objetos, pode-se fazer uso de Padrões de Projeto como diretrizes que contemplam soluções para uma série de problemas comuns. No entanto, há o que se convém chamar de *Anti-padrões de projeto,* que são diretrizes de projeto de *software* orientado a objetos consideradas ruins e que se recomenda evitar.

Dito isso, analise as afirmações abaixo:

I. Deve-se priorizar a composição ao invés da herança.
II. Usa-se uma Interface para especificar um comportamento comum que pode ser implementado de diversas maneiras.
III. Uma classe deve englobar diversas responsabilidades diferentes.
IV. Recomenda-se que um método de um objeto qualquer não deve invocar métodos de objetos que não pertençam ao seu escopo ou ao escopo de sua classe.
V. Recomenda-se a criação de classes cujos objetos têm vida curta, pouca utilização e não têm grandes responsabilidades.

Estão corretas apenas as afirmativas.

a) I e IV apenas .
b) II, IV e V apenas.
c) I, II e III apenas.
d) III e V apenas.

39. Que alternativa apresenta o Padrão de Projeto que tem por objetivo encapsular a criação de objetos, podendo retornar instâncias de objetos de tipos diferentes que herdam de uma classe ou implementam uma interface em comum?

a) Command.
b) Factory.
c) Adapter.
d) Data Access Object (DAO).

40. Qual o Padrão de Projeto que se aplica para utilizar uma classe legada, através de uma nova interface, sem que seja permitido modificar a classe legada?

a) Visitor.
b) Command.
c) Adapter.
d) Decorator.