|  |  |
| --- | --- |
| Faculdade de tecnologia SENAIsc em FlorianópolisCurso: Superior de Tecnologia em Análise e Desenvolvimento de SistemasSemestre: 2013/2 Data: 07/11/13Unidade Curricular: Programação Orientada a Objetos Competência(s): Modelo de Objetos: Classes e Objetos, Abstração X Representação de dados, Atributos, Métodos e Mensagens, Introdução a programação orientada a objetos, Herança, Modificadores de acesso, Construtores, Interfaces, Classes abstratas e Polimorfismo.  Docente: Artur Todeschini Crestani | **Logo SENAI jpeg** |

**AVALIAÇÃO TEÓRICA.**

ALUNO(A):

1. **Indicadores, Critérios e Significados da Avaliação**

|  |  |  |
| --- | --- | --- |
| **Item** | **CRITÉRIOS PARA A NOTA[[1]](#footnote-1)**  (pontuação) | |
|  | **Para obter conceito A o aluno deverá:**  1. Responder 14 das questões teóricas e praticas solicitadas corretamente.  2. A interpretação da questão faz parte da avaliação.  **Para obter conceito B o aluno deverá:**  1. Responder 14 das questões teóricas e praticas solicitadas corretamente.  11 questões.  **Para obter conceito C o aluno desenvolver:**  1. Responder 11 ou menos as questões teóricas e praticas. | |
| **Conceito** | |  |

1. **Atividades da sua avaliação**

|  |  |
| --- | --- |
| **Conceito** | **Significados** |
| **A** | **Competências demonstradas integralmente com excelência**: realiza as atividades propostas, aplicando conhecimentos, habilidades e atitudes exigidas de acordo com os critérios apresentados. Realiza as atividades com autonomia e demonstra domínio das competências (conhecimentos, habilidades e/ou atitudes) que estão sendo avaliadas. Demonstra capacidade de conduzir equipes de trabalho e auxiliar no processo de aprendizagem da mesma. Demonstra, também, capacidade de aplicar, adaptar e articular bases científicas e tecnológicas de forma criativa e inovadora, superando as expectativas em relação aos resultados esperados. |
| **B** | **Competências demonstradas integralmente**: realiza as atividades propostas, aplicando conhecimentos, habilidades e atitudes exigidas de acordo com os critérios apresentados. Realiza as atividades com autonomia e demonstra domínio das competências (conhecimentos, habilidades e/ou atitudes) que estão sendo avaliadas. |
| **C** | **Competências demonstradas parcialmente**: necessita de auxílio ou de acompanhamento para realização das atividades propostas de acordo com os critérios apresentados a fim de demonstrar o desenvolvimento das competências (conhecimentos, habilidades e/ou atitudes) que estão sendo avaliadas. |
| **D** | **Competências não demonstradas**. |

1. Sobre os modificadores de acesso **protected** e **default** (ausência do modificador). Descreva quais as semelhanças e diferença entre desses dois modificadores.
2. Dado o código abaixo: Qual(ais) fragmentos de código abaixo irá compilar? Marque todas as corretas que compilarem.

public interface Base {

boolean m1();

byte m2(short s)

}

1. public interface B2 implements Base { }
2. public abstract class C1 extends Base { }
3. public abstract class C2 implements Base { }
4. public class C3 implements Base { }
5. public classe C4 implements Base {

boolean m1() { return 1; }

byte m2(short s) { return Short.valueOf(s).byteValue() }

}

1. public interface B3 extends Base { }
2. Qual das seguintes afirmativas está correta? (Marque todas as corretas.)
3. “X estende Y” é correto se, e somente se, X for uma classe e Y for uma interface.
4. “X estende Y” é correto se, e somente se, X for uma interface e Y for uma classe.
5. “X estende Y” é correto se X e Y forem ambos classe ou ambos interfaces.
6. “X estende Y” é correto para todas as combinações de X e Y sendo classe e ou interfaces.

1. Descreva o que é encapsulamento. Quais os benefícios do encapsulamento e de exemplos da sua utilização.

1. Sobre relacionamento entre classes descreva com suas palavras o que é um relacionamento “é um” e o que é o relacionamento “tem um”. Considere todos os conceitos estudados. De exemplos e cite qual dos dois é mais flexível e por que.
2. Sobre acoplamento e coesão, quais afirmativas são verdadeiras? (Marque todas as corretas)
3. A coesão é o princípio da programação OO mais intimamente associado com o ato de esconder os detalhes da implementação.
4. A coesão é o princípio da programação OO mais intimamente associado o ato de certificar-se de que as classes só conhecerão outras classes através das suas API’s.
5. A coesão é o princípio da programação OO mais intimamente associado com o ato de certificar se de que a classe foi elaborada com um único e bem focado propósito.
6. A coesão é o principio da programação OO mais intimamente associado com o ato de se permitir que um mesmo objeto seja visto como tendo muitos tipos.
7. Sobre polimorfismo, quais são as afirmativas verdadeiras? (Marque todas as corretas)
8. Polimorfismo significa “muitas formas”.
9. Uma variável de referência é sempre de um só tipo imutável, mas ela pode referir-se a um objeto do subtipo.
10. Um mesmo objeto pode ser referido por variáveis de instâncias de muitos tipos diferentes, desde que elas sejam do mesmo tipo ou de um supertipo do objeto.
11. O tipo da variável de referência (não o tipo do objeto) determina quais métodos poderão ser chamados.
12. As chamadas polimórficas a métodos se aplicam a métodos de instâncias sobrescritos.
13. Em suas palavras explique a diferença de métodos sobrescritos e métodos sobrecarregados. Faça um(dois ou mais) pequeno(s) código(s) que demonstre um pequeno exemplo de método sobrescritos e de métodos sobrecarregado.
14. Sobre o modificador de acesso static (estático). Assinale todas as corretas.
15. Usa-se métodos static para implementar comportamentos que não sejam afetados pelo estado de quaisquer instância.
16. Usa-se variáveis static para armazenar dados que sejam específicos à classe, em vez de à instância – só existirá uma cópia de uma variável static.
17. Todos os membros static (atributos e métodos) pertencem a uma instancia e não a classe.
18. Um método static pode acessar uma variável de instância, pois estão na mesma classe.
19. Métodos static não podem ser sobrescritos, mas podem ser redefinidos.

1. Sobre acoplamento e coesão. (marque todas as questões corretas).
2. O acoplamento refere-se ao nível em que uma classe conhece ou usa membros (atributos e métodos) de uma outra classe.
3. O acoplamento fraco é o estado desejável para classes bem encapsuladas, que minimizam as referências umas às outras e limitam a extensão do uso da API.
4. O acoplamento forte é o estado indesejável de se ter classes que desobedecem às regras do encapsulamento.
5. A alta coesão é o estado desejável de uma classe cujos membros dão suporte a um único e bem definido papel de abstração ou responsabilidade.
6. Dado o código abaixo, responda a questão Está classe está coesa? Justifique sua resposta. (Observação respostas sem justificativas para essa questão não serão consideradas).

public class DemoCoesao {

private Object[] objetos = null;

public void setObjctos(Object[] \_objetos) {

objetos = \_objetos;

}

public Object[] getObjetos() {

return this.objetos;

}

}

1. Sobre o modificador final, explique quais as imprecações do seu uso em atributos, métodos.

1. Sobre interfaces e classes abstract. (Marque todas corretas).
2. Tanto classes abstract e interfaces podem ter construtores.
3. Interfaces e classes abstract, não podem ser instanciadas.
4. Interfaces e classes abstract podem ser utilizadas como tipo de variáveis.
5. Ao sobrescrevermos métodos abstratos de uma interface usa visibilidade pode ser protected, public, default.
6. Classes abstract podem conter métodos que tenham corpo.
7. Interfaces podem conter métodos que tenham corpo.
8. Interfaces possuem construtores.
9. Classes abstract podem definir construtores.
10. O modificador abstract pode ser utilizado em classes, métodos e atributos.

1. Sobre as palavras reservadas do Java this e super, fale qual é a sua implicações dentro do uso de uma classe.
2. Sobre construtores, marque todas as afirmativas corretas.
3. Toda classe tem que ter pelo menos um construtor, se o programador não coloca o compilador coloca um construtor padrão que será public e sem argumentos.
4. Construtor é um método que tem o mesmo nome da classe e tem o seu retorno void.
5. Uma interface tem construtor.
6. Construtores são métodos concretos.
7. Construtores são herdados.
8. Quando utilizamos herança a primeira coisa que temos que fazer é a chamada do construtor da classe pai na primeira linha do construtor da classe filha.
9. Podemos chamar o método construtor através da palavra reservada new.
10. Sobre modificadores de acesso. Marque todas as questões corretas.
11. Há três modificadores de acesso: public, protected e private.
12. Há quatro níveis de acesso: public, protected, (default) ausência de modificador e private.
13. A visibilidade das classes gira em torno de se o código de uma classe pode criar uma instancia de outra classe.
14. A visibilidade das classes gira em torno de se o código de uma classe pode estender (herdar ou criar subclasses) outra classe.
15. A visibilidade das classes gira em torno de se o código de uma classe pode acessar métodos e variáveis de outra classe.

1. [↑](#footnote-ref-1)