Pergunta 1

0,25 em 0,25 pontos

✓ Analise o código abaixo, que define a classe Matemática:

```
public class Matematica {
 public int calculo (int a, int b, int c) {
  //...lógica do cálculo
 }
 public int calculo (int a, int b) {
  //...lógica do cálculo
 }
 public double calculo (double a, double b, double c) {
  //...lógica do cálculo
 }
}
```

O programa acima caracteriza um recurso de O.O. conhecido como:

Resposta Selecionada: ✓ e. Sobrecarga de Métodos.

Respostas:
a. Encapsulamento.
b. Sobrescrita de Métodos.
c. Polimorfismo de Classes.
d. Herança de Classes.
✓ e. Sobrecarga de Métodos.

Feedback da resposta: Resposta: E
Comentário: De acordo com a classe "Matematica" do enunciado, os três métodos apresentados possuem o mesmo nome, porém diferenciados nos parâmetros, Seja na quantidade deles, ou em seus tipos de dados, o que caracteriza uma Sobrecarga de Métodos.

Pergunta 2

0,25 em 0,25 pontos

✓ Analise o código abaixo, que define duas classes envolvendo herança, uma terceira classe que, através de seus métodos, se utiliza das classes anteriores, além de uma quarta classe de teste que as instancia:

```
--------------------------------
public class ClasseA {
 // atributos e métodos da ClasseA
}
--------------------------------
public class ClasseB extends ClasseA {
 // atributos e métodos da ClasseB
}
--------------------------------
public class ClasseC {
 public void metodo01(ClasseA ca) {
  //... lógica do método
 }
 public void metodo02(ClasseB cb) {
  //... lógica do método
 }
 public void metodo03(ClasseC cc) {
  //... lógica do método
 }
 // demais atributos e métodos da ClasseC
}
--------------------------------
public class Teste {
 public ClassA c1 = new ClassA();
 public ClassB c2 = new ClassB();
 public ClassC c3 = new ClassC();
 public void metodoTeste() {
  //invocação dos métodos da ClasseC
 }
}
--------------------------------
```

A partir das opções abaixo, qual delas **não** pode ser inserida no "metodoTeste" da ClasseC, como uma invocação dos métodos da ClasseC?

Resposta Selecionada: ✓ e. c3.metodo02(c1);

Respostas:
a. c3.metodo02(c2);
b. c3.metodo01(c2);
c. c3.metodo01(c1);
d. c3.metodo03(c3);
✓ e. c3.metodo02(c1);

Feedback da resposta: Resposta: E
Comentário: Sabemos que de acordo com o Polimorfismo de Classes, um objeto criado a partir de uma superclasse pode se comportar como um objeto criado a partir de uma das subclasses daquela superclasse (e não vice-versa). De acordo com o enunciado, o objeto c1 é do tipo ClasseA. Ao chamar o metodo02 da ClasseC, o mesmo deve receber como parâmetro um objeto do tipo ClasseB que é uma subclasse da ClasseA. Assim, a opção errada é aquela que chama o método02 enviando como parâmetro um objeto c1 já que uma subclasse não pode se comportar inteiramente como um objeto do tipo de sua superclasse.

Pergunta 3

0,25 em 0,25 pontos

✓ Quanto às Classes Finais e Abstratas podemos afirmar que:

I. As Classes Finais podem ser instanciadas, como também podem ser herdadas.
II. As Classes Finais não podem ser herdadas.
III. As Classes Abstratas são mais genéricas e por isso devem ser herdadas a fim de serem complementadas.
IV. Classes Abstratas só podem possuir métodos abstratos.
V. Classes Abstratas só podem ser instanciadas.

Resposta Selecionada: ✓ b. Somente as alternativas II e III estão corretas.

Respostas:
a. Somente a alternativa II está correta.
✓ b. Somente as alternativas II e III estão corretas.
c. Somente as alternativas I, IV e V estão corretas.
d. Somente as alternativas II e V estão corretas.
e. Somente as alternativas III e IV estão corretas.

Feedback da resposta: Resposta: B
Comentário: Lembrando que classes finais são classes que não podem ser herdadas (como indica a afirmação II), e classes abstratas são classes que não podem ser instanciadas, onde a afirmação III diz que a partir de herança, suas características serão gerais para todas as subclasses, em que cada uma de suas subclasses possuirão características mais específicas.

Pergunta 4

0,25 em 0,25 pontos

✓ Sobre Interfaces, podemos afirmar que:

I. Interfaces são classes finais e abstratas.
II. Nas Interfaces, os métodos são implicitamente abstratos.
III. Nas Interfaces, os atributos são implicitamente estáticos e finais.
IV. Uma interface pode conter vários atributos e métodos.

Resposta Selecionada: ✓ c. Somente as alternativas II, III e IV estão corretas.

Respostas:
a. Somente as alternativas I e II estão corretas.
b. Somente a alternativa IV não está correta.
✓ c. Somente as alternativas II, III e IV estão corretas.
d. Somente as alternativas II e IV estão corretas.
e. Somente a alternativa III está correta.

Feedback da resposta: Resposta: C
Comentário: Algumas características das interfaces estão descritas nas afirmações II, III e IV, de forma que a afirmação I está errada, pois as interfaces não são consideradas classes, mas que suas características se aproximam (até certo ponto) às de uma classe totalmente abstrata.

Pergunta 5

0,25 em 0,25 pontos

✓ Tendo-se os seguintes atributos:

```
public int a = 1, b = 2, c = 3;
public double p = 7, q = 8, r = 9;
public String t = "15";
```

Qual dos seguintes comandos **não** podem ser realizados:

Resposta Selecionada: ✓ e. int x = (int) t;

Respostas:
a. int x = Integer.parseInt(t);
b. double y = (int) p;
c. double y = c + Double.parseDouble(t);
d. float z = (float) q + Float.parseFloat(t);
✓ e. int x = (int) t;

Feedback da resposta: Resposta: E
Comentário: Vimos que para que possamos transformar o valor de uma String em um valor numérico, temos que nos utilizar de métodos das Wrapper Classes, de forma que um simples "casting" não transformaria esse valor string em um valor numérico. Lembrando que um "casting" faz a transformação entre tipos numéricos de categorias diferentes (de maior capacidade numérica para um de menor capacidade numérica).

Pergunta 6

0,25 em 0,25 pontos

✓ Sobre a utilidade das interfaces, podemos afirmar que:

Resposta Selecionada: ✓ a. As interfaces podem garantir, que determinadas regras de negócio sejam implementadas nos sistemas construídos na linguagem Java, obrigando-se a implementação de métodos que definam essas regras ao sistema.

Respostas:
✓ a. As interfaces podem garantir, que determinadas regras de negócio sejam implementadas nos sistemas construídos na linguagem Java, obrigando-se a implementação de métodos que definam essas regras ao sistema.
b. As interfaces protegem os atributos das classes, definindo como eles devem ser populados (como os seus valores devem ser atribuídos).
c. A partir das interfaces observamos, nas classes que as implementam, o que chamamos de sobrecarga de métodos, garantindo os múltiplos usos das suas características.
d. Os métodos implementados diretamente nas interfaces poderão ser invocados pelas classes que as implementam.
e. Ao instanciar uma interface, podemos definir os valores de seus atributos a partir da invocação de seus métodos.

Feedback da resposta: Resposta: A
Comentário: Sabemos que uma interface é um elemento que possui atributos finais e estáticos (e portanto, caracterizando constantes) e/ou métodos abstratos, de modo que esses não possuirão implementações na própria interface, mas sim nas classes que a implementarem. Assim, ao implementá-las nas classes, essas últimas é que deverão conter os métodos implementados daquela interface.

Pergunta 7

0,25 em 0,25 pontos

✓ Analise as afirmações abaixo:

I – Uma classe abstrata deve ser uma classe que contém características mais genéricas.
II – Uma classe final deverá possuir características mais específicas da regra de negócio a qual está sendo definida.
III – Um método final de uma superclasse, para que adquira as características específicas da subclasse, deverá ser substituído na subclasse por um outro método contendo a mesma assinatura.
IV – Métodos finais impedem a sobrecarga de métodos em subclasses.

De acordo com as afirmações acima, estão corretas:

Resposta Selecionada: ✓ a. Apenas as afirmações I e II.

Respostas:
✓ a. Apenas as afirmações I e II.
b. Apenas as afirmações III e IV.
c. Apenas a afirmação II.
d. Apenas a afirmação IV.
e. Apenas as afirmações I e III.

Feedback da resposta: Resposta: A
Comentário: Classes finais são aquelas que não podem mais ser herdadas, de forma que ela já contém os métodos que a caracterizam, e, portanto, são métodos considerados mais específicos, já que não existirão subclasses especificando ainda mais suas características. Já as classes abstratas, que não podem ser instanciadas, podem ser herdadas por outras classes, e com isso, os métodos daquela classe abstrata serão métodos "gerais" pertencentes a todas às suas subclasses.

Pergunta 8

0,25 em 0,25 pontos

✓ Qual a utilidade do método "parse" das "Wrapper Classes" representantes dos tipos primitivos numéricos?

Resposta Selecionada: ✓ d. Retornar um valor numérico a partir de um valor de uma string, permitindo que com ele se realize cálculos numéricos.

Respostas:
a. Alterar a tipagem da variável do tipo String, de forma a se tornar uma variável de um tipo numérico.
b. Verificar se um valor é igual a um valor de outra variável.
c. Verificar se um determinado valor numérico é um valor par (divisível por dois).
✓ d. Retornar um valor numérico a partir de um valor de uma string, permitindo que com ele se realize cálculos numéricos.
e. Realizar a troca de informações entre variáveis.

Feedback da resposta: Resposta: D
Comentário: Vimos que os tipos numéricos primitivos, inteiros e reais (short, int, long, float, double) possuem uma classe que a representa, uma "Wrapper Class", e que possui um método "parse" que recebe em seu parâmetro um valor do tipo String, e retorna um valor numérico no tipo primitivo que representa. Precisa-se apenas tomar cuidado com valores "strings" que não podem se transformar em número (como letras, símbolos e palavras).

Pergunta 9

0,25 em 0,25 pontos

✓ Imagine que uma equipe esteja trabalhando na confecção de um game de corridas de carro na linguagem Java, onde foi criada uma classe Carro que será uma classe que conterá todas as características gerais de todos os "carros-personagens" do jogo, e que, portanto, será herdada por todas as classes que representarão esses personagens. O coordenador da equipe de desenvolvedores pretende exigir que em todas as classes de "carros-personagens", contenham métodos de controle individuais e próprios, e que deverão ser implementados por cada um dos desenvolvedores de sua equipe ao criarem uma classe de "carro-personagem", cada uma com suas particularidades individuais. Quanto a esses métodos de controle, qual o recurso de programação que permite a forma mais organizada dessa exigência ocorrer automaticamente durante o desenvolvimento da aplicação (do jogo)?

Resposta Selecionada: ✓ b. Tornar a classe Carro abstrata, e definir esses métodos de controle como métodos abstratos na classe Carro.

Respostas:
a. Encapsular as classes "carro-personagens", e controlar seus atributos através dos métodos *setters* e *getters*.
✓ b. Tornar a classe Carro abstrata, e definir esses métodos de controle como métodos abstratos na classe Carro.
c. Implementar os métodos de controle na classe Carro, e depois deixar que esses métodos sejam sobrescritos.
d.
Criar todos os métodos de controle (para todos os carros-personagens, que serão subclasses da classe Carro) dentro da própria classe Carro, gerando uma sobrecarga de métodos, e depois invocando convenientemente cada um deles, dependendo do carro-personagem que esteja sendo controlado.
e. Deixar que os programadores de cada um dos carros-personagens, por sua conta e risco, perceba que esses métodos precisam ser implementados.

Feedback da resposta: Resposta: B
Comentário: Os métodos abstratos são aqueles que não possuem implementação, que existem numa classe obrigatoriamente abstrata, e que exigem sua implementação nas classes filhas. Desta forma, utiliza-se classes abstratas com métodos abstratos para também alertar aos Desenvolvedores sobre a necessidade da criação daqueles métodos nas classes (subclasses) que herdarem aquela classe (superclasse).

Pergunta 10

0,25 em 0,25 pontos

✓ Ao se criar um programa desktop em linguagem Java, qual dos seguintes métodos permite a entrada de dados a partir de uma tela de diálogo?

Resposta Selecionada: ✓ a. método "showInputDialog" da classe "JOptionPane".

Respostas:
✓ a. método "showInputDialog" da classe "JOptionPane".
b. método "showConfirmDialog" do objeto "JOptionPane".
c. método "showMessageDialog" do objeto "JOptionPane".
d. método "insert" do objeto "in" da classe "System".
e. método "input" da classe "System".

Feedback da resposta: Resposta: A
Comentário: Em um programa desktop, utilizamos o método "showInputDialog(...)" da classe "JOptionPane", que é uma caixa de diálogo que contém um campo texto para entrada de dados, já que esse método retorna um valor do tipo String. Para isso, fazemos com que uma variável receba o retorno deste método, que será o texto descrito pelo usuário naquele campo.