Exercícios de fixação de conteúdo

Parte I

1. Escreva uma classe em Java de nome PSlogam, cujo método main é responsável por imprimir na tela o texto "Bem vindo a programação Java".

2. Escreva uma classe em Java de nome Intervalo, cujo método main é responsável por imprimir na tela todos os números pares de 299 a 101 em ordem decrescente.

3. Escreva uma classe em Java de nome Inverso, cujo método main é responsável por receber uma palavra ou frase e exibi-la em caracteres maiúsculos e invertida. Exemplo: para a entrada POLIMIG, exibir a saída GIMILOP.

4. Escreva exatamente a saída da execução do método main da classe Operacoes:

```
public class Operacoes {
	int a, b;

	void alteraValores(int n1, int n2) {
		a = n1;
		b = n2;
	}

	int operacao1() {
		return 2 * a + 2 * b;
	}

	int operacao2() {
		return a * b;
	}

	public static void main(String[] args) {
		Operacoes obj = new Operacoes();
		obj.alteraValores(10,20);
		System.out.println(obj.operacao1());
		System.out.println(obj.operacao2());
	}
}
```

5. Uma classe Lampada com atributo ligada (tipo boolean) e métodos liga() e desliga() que nada retornam. O método liga torna o atributo ligada true e o método desliga torna o atributo ligada false. Crie também o método observa() que retorna a String “ligada” se a lâmpada estiver ligada e “desligada” se a lâmpada estiver desligada.

6. Acrescente um método main à classe Lampada da questão anterior. O método main deve:

a) Instanciar 2 objetos do tipo Lampada (lamp1 e lamp2)

b) Ligar o objeto lamp1 e desligar o objeto lamp2.

c) Usar o método observa() para mostrar se os objetos lamp1 e lamp2 estão “ligados” ou “desligados”.

7. Transcreva os códigos gerados nas questões 5 e 6 para a linguagem de programação C++

8. Escreva uma classe **Quadrado** com atributo **lado** do tipo **double**. A classe deve ter um construtor que recebe como parâmetro o **lado** do quadrado. Deve também ter os métodos **area()** e **perimetro()** que retornam respectivamente a área e o perímetro do quadrado, cujas fórmulas são as seguintes:
area = $lado^2$
perimetro = 4 x lado

9. Escreva uma classe **UsaQuadrado** cujo método **main** instancia os 3 objetos a seguir:

q1 lado: 2	q2 lado: 4	q3 lado: 5

Observe que os 3 objetos são instâncias da classe **Quadrado** criada na questão anterior. Após instanciar os 3 objetos, o método **main** ainda deve mostrar a área e o perímetro dos 3 quadrados instanciados.

10. Escreva uma classe em Java chamada **Estoque**. Ela deverá possuir:
a) os atributos **nome** (String), **qtdAtual** (int) e **qtdMinima** (int).
b) um construtor sem parâmetros e um outro contendo os parâmetros nome, qtdAtual, e qtdMinima.
c) os métodos com as seguintes assinaturas:

```
void mudarNome(String nome)
void mudarQtdMinima(int qtdMinima)
void repor(int qtd)
void darBaixa(int qtd)
String mostra()
boolean precisaRepor()
```

Os atributos **qtdAtual** e **qtdMinima** jamais poderão ser negativos. O método repor aumenta qtdAtual de acordo com o parâmetro qtd. O método darBaixa diminui qtdAtual de acordo com o parâmetro qtd. O método mostra() retorna uma String contendo o nome do produto, sua quantidade mínima, sua quantidade atual. O método precisaRepor retorna true caso a quantidade atual esteja menor ou igual à quantidade mínima e false, caso contrário.

11. Escreva uma classe **UsaEstoque** cujo método **main** instancia os 3 objetos a seguir:

estoque1 nome: Impressora Jato de Tinta qtdAtual: 13 qtdMinima: 6	Estoque2 nome: Monitor LCD 17 polegadas qtdAtual: 11 qtdMinima: 13	estoque3 nome: Mouse Ótico qtdAtual: 6 qtdMinima: 2

Depois disso, execute as seguintes operações na seguinte ordem:
a) Dar baixa em 5 unidades de estoque1.
b) Fazer a reposição de 7 unidades de estoque2.
c) Dar baixa em 4 unidades de estoque3.
d) Exibir a saída do método precisaRepor dos 3 objetos.
e) Exibir a saída do método mostra para apresentar as informações sobre os 3 objetos.

12. Escreva as seguintes classes:
a) Uma classe **Pessoa** atributos **nome** (tipo String) e **sobrenome** (tipo String). Cada um desses

atributos deve ter métodos para lê-los e alterá-los (getters e setters). A classe **Pessoa** ainda deve ter um método chamado **getNomeCompleto** que não possui parâmetros de entrada e que retorna a concatenação do atributo **nome** com o atributo **sobrenome**. Além disso, a classe deve possuir um **construtor** sem parâmetros e um outro **construtor** que recebe como parâmetros o nome e o sobrenome da pessoa e altera respectivamente os atributos nome e sobrenome.

b) Uma subclasse de Pessoa, chamada **Funcionario**. A classe Funcionario deve ter os atributos **matricula** (tipo int) e **salario** (tipo double), com seus respectivos métodos para leitura e alteração (getters e setters). O salário de um funcionário jamais poderá ser negativo. Todo funcionario recebe seu salario em duas parcelas, sendo 60% na primeira parcela e 40% na segunda parcela. Assim, escreva os métodos **getSalarioPrimeiraParcela** que retorna o valor da primeira parcela do salário (60%) e **getSalarioSegundaParcela** que retorna o valor da segunda parcela do salário (40%).

c) Uma subclasse de Funcionario, chamada **Professor**. Todo professor recebe seu salário em uma única parcela. Assim, deve-se sobrescrever os métodos **getSalarioPrimeiraParcela** e **getSalarioSegundaParcela**. O método **getSalarioPrimeiraParcela** da classe Professor deve retornar o valor integral do salário do professor e o método **getSalarioSegundaParcela** do professor deve retornar o valor zero.

d) Uma classe **UsaPessoa** que instancia os seguintes objetos:

pessoa1 (Pessoa) nome: Mario sobrenome: Lopes	pessoa2 (Funcionario) nome: Lucas sobrenome: Mendes salario: 2000.00	pessoa3 (Professor) nome: Rafael sobrenome: Lira salario: 500.00

Depois disso, execute as seguintes operações na seguinte ordem:

a) Exibir a saída do método getNomeCompleto para os 3 objetos.

b) Exibir a saída dos métodos getSalarioPrimeiraParcela e getSalarioSegundaParcela para os objetos pessoa2 e pessoa3

13. Implemente as seguintes classes:

a) Implemente uma classe Equipamento com o atributo ligado (tipo boolean) e com os métodos liga e desliga. O método liga torna o atributo ligado true e o método desliga torna o atributo ligado false.

b) Implemente uma classe EquipamentoSonoro que herda as características de Equipamento e que possui os atributos volume (tipo short) que varia de 0 a 10 e stereo (tipo boolean). A classe ainda deve possuir métodos para ler e alterar o volume (getter e setter), além dos métodos mono e stereo. O método mono torna o atributo stereo falso e o método stereo torna o atributo stereo verdadeiro. Ao ligar o EquipamentoSonoro através do método liga, seu volume é automaticamente ajustado para 5.

14. Implemente: a) Uma classe Transporte com atributos ligado (tipo boolean) e velocidade (tipo int) e métodos liga() e desliga(). O método liga torna o atributo ligado true e o método desliga torna o atributo ligado false, além de tornar a velocidade zero. Crie também métodos get/set para modificar o atributo velocidade, sendo que a velocidade não pode ser negativa.

b) Uma subclasse de Transporte chamada Carro. Carro deve ter o atributo quilometragem (tipo int) e os métodos necessários para lê-lo e alterá-lo (get/set). A quilometragem não pode ser negativa, nem ultrapassar o valor 999999. A velocidade do Carro não pode ser negativa, nem ultrapassar 200.

15. Implemente: a) Uma classe Conta com atributo saldo e métodos depositar e sacar para diminuir e aumentar o atributo saldo, respectivamente, a partir de um valor especificado. O atributo saldo pode ser negativo.

b) Implemente uma subclasse de Conta chamada Poupanca. Poupanca deve ter o atributo diaRendimento do tipo int que armazena o dia do mês em que ocorre o rendimento da poupança. Além disso, ainda deve possui os métodos necessários para ler e alterar o atributo diaRendimento. O atributo saldo da Poupanca não pode ser negativo.

PARTE II

1.Dado o seguinte código:

```
interface Simples {
        boolean mostra( );
        byte ver(short s);
}
```

Qual fragmento de código irá compilar? (Marque todas as corretas)

A. interface Simples2 implements Simples { }
B. abstract class Class2 extends Simples {
    public boolean mostra( ) { return true; } }
C. abstract class Class2 implements Simples { }
D. abstract class Class2 implements Simples {
    public boolean mostra( ) { return (true) ; } }
E. class Class2 implements Simples {
    boolean mostra( ) {return false; }
    byte ver(short s) { return 42; } }

2. Qual das seguintes opções declara uma classe abstract compilável? (Marque todas as Corretas.)

A. public abstract class Canine { public String speak( ); }
B. public abstract class Canine { public String speak ( ) { } }
C. public class Canine { public abstract String speak ( ) ; }
D. public class Canine abstract { public abstract String speak ( ) ; }

3. Qual afirmativa está correta?

A. “X estende Y” é correto se, e somente se, X for uma classe e Y for uma interface.
B. “X estende Y” é correto se, e somente se, X for uma interface e Y for uma classe.
C. "X estende Y" é correto se X e Y forem ambos classes ou ambos interfaces.
D. "X estende Y" é correto para todas as combinações de X e Y sendo classes e/ou interfaces.

4. Quais das seguintes são declarações válidas? (Marque todas as corretas.)

A. int $x;
B. int 123;
C. int _123;
D. int #dim;
E. int *divide;
F. int central_sales_region_Summer_200S_gross_sa1es

5. Quais das seguintes opções são declarações válidas? (Marque todas as corretas.)

A. short x [ ];
B. short [ ] y;
C. short [5] x2;
D. short z2 [5] ;
E. short [ ] z [ ] [ ];
F. short [ ] y2 = [5] ;

6. Quais afirmativas são verdadeiras? (Marque todas as corretas)

A. Os relacionamentos Tem-Um sempre dependem da herança.
B. Os relacionamento Tem-Um sempre dependem das variáveis de instâncias.
C. Os relacionamento Tem-Um sempre precisam de pelo menos dois tipos de classes.

D. Os relacionamento Tem-Um sempre dependem do polimorfismo.

7. Usando os fragmentos abaixo, complete o seguinte código de forma que ele compile. Observação, talvez não seja preciso preencher todos os espaços em branco.

Código:

```
class Agedp{
__________  __________  __________  __________  __________
  public Agedp(int x) {
        __________  __________  __________  __________  __________
  }
}

public class Kinder extends Agedp {
__________  __________  __________  __________  __________  __________
  public Kinder (int x) {
        __________  __________  __________  __________  __________ ( );
  }
}
```

Fragmentos: Use cada um dos seguintes fragmentos quantas vezes for preciso, ou deixe sem uso:

Agedp	super	this	
(	)	{	}
;			