DISCIPLINA...: Programação Orientada a Objetos
PROFESSOR...: José Antonio Gonçalves
ALUNO...: Jean Alves Rocha **RA...:** 2313057

1º Questionário

1) Por que falamos que Java é totalmente aderente às técnicas de Orientação a Objetos?
R: Java é aderente por diversos fatores, como por exemplo, os tipos de dados classes, que podem conter estruturas parecidas com funções chamadas métodos, e “variáveis” denominadas atributos. Através das classes, pode-se instanciar objetos – Materialização da Classe – que é advinda do processo de instanciação. Isso e outros fatores como os conceitos de Polimorfismo, herança e encapsulamento, que são do paradigma de POO, fazem Java ser aderente.

2) Explique o que é e como se utiliza o processo de “abstração”.
R: Abstração é a capacidade de representar uma entidade do mundo real na computação que foca apenas nos pontos que são de interesse de quem implementará. O processo é feito por meio de levantamento de dados.

3) Quais são os artefatos produzidos na programação orientada a objetos?
R: Classes, objetos, métodos e atributos.

4) O que é um “tipo primitivo” de dados?
R: Tipo primitivo de dados são dados básicos fornecidos por linguagens de programação como construção básica.

5) O que é um “tipo abstrato” de dados?
R: Tipo abstrato de dados (TAD) pode ser visto como um modelo matemático que encapsula um modelo de dados e um conjunto de procedimentos que atuam com exclusividade sobre os dados encapsulados.

6) Explique o que é o “Garbage Collector”. Como este recurso pode dinamizar o funcionamento do sistema?

R: É um coletor de lixo que “limpa” da memória principal os objetos que não estão sendo mais utilizados, devido a isso o recurso dinamiza o funcionamento do sistema, focando apenas no que está de fato sendo utilizado.

7) Considerando o **modo Shell** (linhas de comando) do sistema operacional Windows, como se faz para:

7.a) Compilar um código fonte java:

R: Usa o cd mais o nome da pasta que estiver até chegar na pasta onde o arquivo .java está e usa o comando **javac nome_do_arquivo**.

7.b) Fazer com que a J.V.M (Máquina Virtual Java) execute uma aplicação.

R: (Estar na pasta que o .class estiver) e usar o comando **java nome_do_arquivo** apenas isso, sem o .class ou .java no final do nome.

8) O que é o “ByteCode”?

R: É um arquivo intermediário entre o baixo nível e alto nível que é rodado através da J.V.M.

9) Explique o que é a característica “Portabilidade”. Como isto é possível com aplicações Java? Para esta resposta relacione 4 “personagens” deste cenário: o código fonte (arquivo .java), o byteCode (arquivo .class), o Sistema Operacional e a JVM (Java Virtual Machine).

R: Cada S.O tem sua estrutura lógica que organiza seu sistema operacional, portanto, quando é feita a compilação de programas geralmente nele é imbutido propriedades características do S.O que ele está sendo rodado. Na linguagem Java, por outro lado, é criado o .java e após sua compilação é gerado o .class, que é denominado como byteCode, este arquivo é rodado através da JVM, a qual é uma intermediária entre a aplicação e o S.O. Por existir um intermediário, o byteCode não precisa ter propriedade do S.O que está sendo gerado, ficando a cargo da JVM, logo o byteCode se torna “genérico” para sistema operacionais, portanto, poderá utilizá-lo em qualquer outro sistema, bastando apenas ter o

intermediário instalado na máquina. Isso é definição de portabilidade: A capacidade de “rodar” em qualquer S.O.

10) Justifique a afirmação que diz que “a segurança em java se dá em dois níveis: proteção de hardware e proteção de software”.
R: Proteção do Hardware (proteção da RAM): Pelo fato de Java não implementar “ponteiros”, garante a integridade no gerenciamento da memória principal.
Proteção de software: Há várias API’s (interfaces para Programação de Aplicações) que são fornecidas nas bibliotecas nativas de Java que reduzem a margem de erro. Além disso, o padrão metodológico de encapsulamento, métodos getters e setters, entre outros, elevam a segurança do software.

11) Explique como aplicamos o conceito de “Modularidade” em Java. Na resposta desta questão deve-se tratar dos conceitos sobre “Acoplagem” e “Coesão”.
R: A aplicação da modularidade se dá na fragmentação dos processos que podem ser em níveis, ou não, definidos pelas classes e que “dialogam” através da sua materialização, os objetos. Dada a forma do processo, pode-se implicar caracterizações do mesmo, as quais são definidas em Acoplagem e Coesão. Essas características são uma inverso da outra, em outras palavras, quando maior a coesão, isto é, há poucos objetos na aplicação, menor a acoplagem.

11. a) Como esta característica pode ajudar na questão da “Manutenibilidade”?
R: Devido a modularidade o processo de manutenção pode ser direciona à um fragmento específico (módulo), agilizando e tornando eficaz o processo de manutenção.

12) Para que serve os objetos:
12. a) this;
R: Especifica um membro da classe (atributo ou método).
12. b) super;
R: Modifica a hierarquia de métodos com assinaturas iguais entre classes mães e filhas dando prioridade à mãe (superclasse).

13) Usando Java, dê um exemplo que contemple as respostas das questões 12.a e 12.b.

```
this.cpf = cpf;
super.calcularSalario();
```

14 - Dentre os conceitos de sustenta a Orientação a Objetos, explique:

14.a) Encapsulamento:

R: Maneira de gerenciar o diálogo entres os objetivos: Aumenta restrição, ou não, das conversas. Em outras palavras, reduz, ou não, a capacidade de gerenciamento dos atributos, métodos, entre outros.

14.a.i) Seus níveis (explique cada um dos três níveis);

R:

Público: Qualquer pessoa com acesso ao .class, que saiba o que a classe contém, poderá modificar os atributos no objeto diretamente.

Privado: Só quem tem acesso à classe poderá modificá-lo diretamente, sendo assim, quando há a materialização da classe, isto é, é feito o objeto através do método construtor, a mudança nos atributos só será feita por meio dos métodos pré-estabelecidos, os públicos. Portanto, privado é aquilo que só é possível acessar por intermédio dos métodos, por padrão serão os métodos getters e setters.

Protegido: Tem acesso ao método e atributos apenas a classe que os detém e as suas especializações, classes filhas.

14.a.ii) Como o Encapsulamento pode nos ajudar na padronização, segurança e manutenibilidade no desenvolvimento de sistemas;

R: Como já descrito acima, a utilização de atributos privados faz com que quem for utilizar o objeto siga padrões pré-estabelecidos por quem o fez, padrões estes como os métodos getters e setters, logo, essa restrição faz que o outro programador, ou quem quer que seja, siga um padrão, portanto a segurança e padronização advém disso. No mais, por existir padronização, a manutenção se torna mais fácil.

14. b) Herança:

R: Faz com que as classes filhas herdem atributos e métodos da classe mãe.

14. b.i) Explique os conceitos “Generalização” e “Especialização”;
R: Quando há herança, a classe mãe, que provém atributos e métodos para as filhas, da perspectiva das filhas, é vista como uma classe genérica, é daqui que vem o conceito de generalização. A especialização é o contrário, é a perspectiva da classe mãe, genérica, para as filhas, que provém seus atributos e métodos e os métodos e atributos delas.

14. b.ii) Como o mecanismo de Herança pode nos ajudar na padronização, segurança e manutenibilidade no desenvolvimento de sistemas;
R: Como as classes podem herdar características de outras classes, isso facilita a padronizar um programa de modo que ele fica mais fácil de ser lido, mantido e seguro, já que vão ter classes que só podem ser acessadas através de outras por conta da herança.

14. c) Polimorfismo:
R: Capacidade de se comportar de maneira distinta.

14. c.i) Sobrecarga;
R: O polimorfismo por sobrecarga é a variação de métodos com mesma assinatura dentro de uma mesma classe.

14. c.ii) Sobrescrita;
R: O polimorfismo por sobrescrita é sobrescrever o um método. Exemplo, existe um método da classe mãe com uma assinatura ‘x’, e na sua classe filha é feito um outro método com a mesma assinatura. Quando o dialogo entre os objetos começar e o método x for chamado, a prioridade será da classe filha, ou seja, houve a sobrescrita do método.

14. c.iii) Coerção;
R: O polimorfismo por coerção é a declaração de um objeto ‘x’ e instanciá-lo como ‘y’. Em outras palavras, é ser algo e se comportar diferente.

15) Construa um programa para exemplificar as respostas das questões 14.a, 4.b e 14.c;

```
abstract class Mamifero{
        //Polimorfismo por sobrecarga: repetição da assinatura.
        public int obterCotaDiariaLeite();
        public int obterCotaDiariaLeite(int peso);

}

//Herança
public class Elefante extends Mamifero{

        //Polimorfimo por sobrescrita
        public abstract int obterCotaDiariaLeite(){
                return 20;
        }
}
public class Principal{
        public static void main(String arg){
                //Polimorfimo por coerção
                Mamifero m = new Elefante();
                System.out.println("Cota diaria de leite: "+m.obterCotaDiariaLeite());
        }


}
```

16) Explique o que são trocas de mensagens? Como isso acontece?

R: Troca de mensagens é o nome que se dá a comunicação entre objetos de diversos tipos, inclusive iguais. Isso acontece através da instanciação dos mesmos e das chamadas métodos, passagens de valores, entre outros.

17) O que é um método construtor? Qual sua importância? Faça um código que demonstre sua explicação.

R: Métodos construtores são métodos especiais que inicializam os objetos. Ele é importante, pois além de instanciar o objeto declarado, ele também pode definir como será feito.

Exemplo:

```
public class calcular{

        private string nome;
        private float bonus;

        public Calcular(){
                nome = “”;
                bonus = 0;
        }
        public Calcular(String nome, float bonus){
                this.nome = nome;
                this.bonus = bonus;
        }
}//fim da classe
```

18) Explique o que são como e quando utilizamos:

18. a) Classe abstrata;

R: É uma classe feita para ser herdada, isto é, ela não poderá ser instanciada. É utilizada para garantir que instanciação só aconteça com as filhas.

18. b) Método abstrato;

R: Método feito dentro da classe abstrata e se faz obrigatório a sobrescrita nas classes filhas. É utilizando quando se quer obrigar o desenvolver a implementar a classe.

18. c Classe final**;**

R: É uma classe que não pode ser herdada. É utilizada quando a classe, por exemplo, é uma filha e dentro da proposta de implementação ela não será herdada. Isso aumenta a segurança do projeto.

18. d) Atributo final;

R: É um atributo constante. É utilizado quando se quer uma constante.

18. e) Método final;

R: É um método que não pode ser sobrescrito, ele é o inverso do abstract. É utilizado quando o desenvolver quer garantir que o outro desenvolvedor, caso queira utilizar o método, utilizará do jeito que o primeiro quis.

19) Dentro da tecnologia Java, explique o que é a estrutura de dados “Interface”. Quando a utilizamos?

R: A estrutura de dados Interface é uma forma de suprir a falta de herança múltipla, onde os métodos, que só são descritas suas assinaturas dentro da classe, obrigam o desenvolvedor a implementá-los. Comporta-se como um método abstrato. Seus atributos se comportam como constantes.

É utilizada quando há a necessidade de implementar herança múltipla.