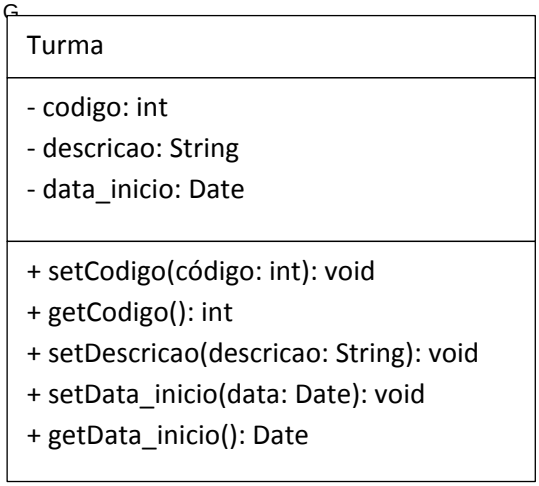
Senac	ORIENTAÇÃO A OBJETOS APLICADA A LINGUAGEM JAVA	Ótimo	Bom	Suficiente	Insuficiente

Instrutor: Wanderlei Silva do Carmo Data:

Aluno:

1 - Analise o digrama e crie código Java correspondente:

G




```
public class Turma{
 private int codigo;
 private String descricao;
 private Date data_inicio;

  public void setCodigo(int  cod){
    this.codigo = cod;
  }
  public int getCodigo(){
     return this.codigo;
  }
  public void setDescricao(String desc){
    this.descricao = desc;
  }
  public void setData_inicio(Date d){
     this.data_inicio = d;
  }
  public Date getData_inicio(){
   return this.data_inicio;
  }
}
```

2 - Sobre o paradigma da orientação a objetos em Java é correto afirmar:

A (x) – Classe abstrata e Interface são utilizadas para obter o polimorfismo
B (x) - Classes abstratas possuem métodos com assinatura e corpo
C (x) – Classes abstratas não podem ser instanciadas e sim herdadas
D (x) - Interfaces devem ser construídos contendo apenas sua assinatura
E (x) - Toda classe que implementa uma ou mais interfaces devem implementar todos os seus métodos.

3 – São tipos de classes:

A ( ) - Interface e Final
B (x) - Concreta e Abstrata
C ( ) – Abstrata e Lógica
D ( ) – Real e Abstrata

4 – Criação do objeto pela alocação de memória para armazenar informações sobre ele. É uma referencia a um endereço de memória onde conterá um objeto, são clássicas definições sobre:

A (x) - Instância
B ( ) – Objeto
C ( ) – Atributo
D ( ) – Método

5 – Para instanciar uma classe usamos a palavra reservada:

A ( ) – private static Class XXX { }
B ( ) – public class XYZ;
C (x) - new
D ( ) - protected

6 – É correto afirmar que: (pode conter ou não mais de uma resposta)

A ( ) – Um objeto é o mesmo que uma classe
B ( ) – Uma classe só existe na memória principal enquanto o programa estiver sendo executado
C (x) – Um objeto possui atributos, métodos e comportamentos
D ( ) - Um objeto pode ser instanciado
E (x) – Um objeto é uma instancia de uma classe

7 - Como é feito a comunicação entre os objetos na memória?

A ( ) - Por meio dos seus atributos
B ( ) – Por meio de suas instancias
C (x) – Por meio de troca de mensagens através de seus métodos.
D ( ) - Criando objetos de um mesmo tipo e referenciando o endereço de memória

8 - Quando um atributo de uma classe deve estar visível e acessível para qualquer outra é comum utilizarmos modificadores. Qual modificador deve ser utilizado para este fim?

A ( ) – private
B ( ) – protected
C ( ) – Friendly
D (x) – public

9 – Qual a diferença básica entre escopo de classe e escopo de instância e qual modificador deve ser utilizado para o escopo de classe?

Escopo de classe os atributos e métodos pertencem a classe enquanto que no escopo de instância atributos e métodos pertencem ao objeto. (atributos e métodos estáticos) - deve ser utilizado o modificador 'static'.

10 - Defina encapsulamento.
Encapsulamento é uma técnica que permite "esconder" os atributos tornando-os privados, ou seja, seus atributos são acessados apenas pelo próprio objeto ou instância, e o acesso à eles somente serão permitidos por meio de seus métodos públicos.

11 – Quais os qualificadores para definição de escopo eu uma classe?
private, public e protected

12 – O que são Construtores e Construtores Sobrecarregados ?
Construtores são métodos que inicializam atributos de uma classe durante sua instanciação e quando possuem várias assinaturas são chamados construtores sobrecarregados, ou seja, para cada assinatura pode-se ter parâmetros diferentes. Ex.: Pessoa p = new Pessoa(); Pessoa p = new Pessoa("Wanderlei",52); Pessoa p = new Pessoa("Wanderlei"). Neste exemplo temos três assinaturas para o construtor. Um construtor é como um método com o mesmo nome da classe.

13 - O que são pacotes na linguagem de programação Java ?
Pacotes são utilizados para organizar classes e outros componentes necessários para o desenvolvimento de aplicações. Bibliotecas, Imagens e classes internas do Java e de terceiros são distribuídas organizadas em pacotes. Pacotes são como pastas.

14 - Marque com G - quando observar uma generalização e E – quando observar uma Especialização nos casos abaixo:

A (G) - class Veiculo {}
B (E) – class Automovel extends Veiculo {}
C (G) – class Pessoa{}
D (E) - class Bicicleta extends Veiculo implements IVeiculoTerrestre
E (E) - class PessoaFisica extends Pessoa{}

15 - Quando uma classe herda características de outra, tais como (atributos e métodos) no paradigma da orientação a objetos damos o nome:

A ( ) - Polimorfismo
B ( ) – Encapsulamento
C ( ) - Superclasse ou ancestral
D (x) – Herança

16 – Quando uma classe é utilizada para uma especialização de uma outra a classe genérica passa a ser chamada:

A ( ) - Polimorfismo
B ( ) – Encapsulamento
C (x) - Superclasse ou ancestral
D ( ) – Herança

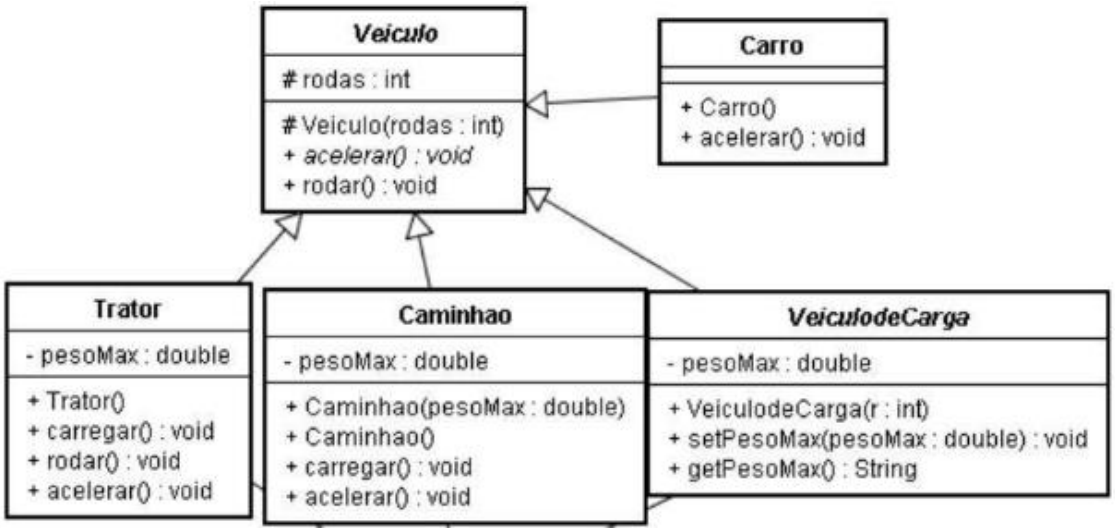
17 - Cite algumas características de uma Interface no contexto da orientação a objetos em Java.

Interfaces somente contém assinaturas de métodos e as classes que a implementam também devem implementar seus métodos. É utilizada em polimorfismo pois deverão ser implementadas e forma diferentes de acordo com a operação desejada na classe que a implementou.

18 - Complete a sentença abaixo conforme o contexto:

A - public class Veiculo extends Terestre implments IVeiculoTerrestre{ ... }
B - public class Crud implements ICrud { ... }
C - Bicicleta bicicleta = new Bicicleta();
D - public void Listar() (não retorna resultados)
E - public List<Produto> Listar() (retorna uma lista de produtos - use uma das coleções)
F - public Produto Exibir() (retorna um único produto)

19 - Crie as classes utilizando os conceitos apreendidos durante nosso curso conforme o diagrama abaixo:



20 - O padrão MVC é um padrão de desenvolvimento de software cuja principal vantagem é a separação entre três principais camadas: dados, negocio e apresentação. Qual o significado de cada letra em MVC e suas características básicas.

M - Model – camada de dados (acesso e persistência)

V - View – camada de apresentação (GUI's)

C - Controller – camada de negócios.
Ligação entre a camada Model e camada View

```
public class Veiculo{
  protected int rodas;
  protected Veiculo(int rodas){}
  public void acelerar();
  public void rodar();
}
public class Carro extends Veiculo {
   public void Carro(){}
}


public class Caminhao  extends Veiculo {
  private double pesoMaximo();


}
//construtor sobrecarregado
 public Caminhao(double pesoMaximo){
   this.pesoMáximo = pesoMaximo;
  }
//Construtor
public Caminhao(){
public void carregar(){ }
public void aceleremar(){ }

}
public class VeiculoCarga  extends Veiculo {
  private double pesoMaximo;
//Construtor
public VeiculoCarga(int r){}
}
```