* **Problema 01**

algo ou um objeto que representa algo do mundo real.

* **Problema 02**

Características: Ano, Modelo, Marca.

Métodos: Desliga, Ligar, Acelerar.

* **Problema 03**

Características são os Atributos.

Comportamento são os Métodos.

* **Problema 04**

Diminuir o programa deixando-o, mas complexo para a programação, possibilitando a reutilização do código.

Utilizando objetos para se comunicarem entre si, para a construção do software de modo geral.

* **Problema 05**

Abstração é dar atenção a aquilo que é realmente importante, abstrair é definir o que é importante para

ser usado em uma classe. Por ex: ao tratar de uma conta bancária, saber o que é importante extrair disso,

num contexto de banco não é interessante saber o tipo de gene de um ser vivo, já se tratando de um sistema de banco de gene,

é algo avançado ou modificado.

* **Problema 06**

Uma classe é uma estrutura de dados que busca representar um objeto,

compreendendo as características e comportamentos dos objetos.

* **Problema 07**

Utilizar uma palavra, começando sempre com uma letra maiúscula, sempre que houver outra palavra,

deve iniciar com letra maiúscula também, não usar espaço, nem iniciar a variável com número,

no singular. Padrão camelCase. Ex: casa, casabairro.

* **Problema 08**

CarroEletrico.

* **Problema 09**

Utilizar uma palavra, começando sempre com uma letra minúscula, sempre que houver outra palavra,

deve iniciar com letra maiúscula também, não usando espaço, nem iniciando a variável com número,

podendo iniciar com $ ou \_ (underscore), mas não é o recomendado. Padrão camelCase. Ex: corDoCabelo.

* **Problema 10**

corDeFundo.

* **Problema 11**

Utilizando uma palavra, começar sempre com uma letra minúsculo,

sempre que houver outra palavra, deve iniciar com letra maiúscula também,não usar espaço, nem iniciar a variável com número,

sendo um verbo. Padrão camelCase. Ex: getCorDoCabelo.

* **Problema 12**

estaVazio

* **Problema 13**

Objeto é uma referência de memória para a instância de uma classe.

Ela passa a existir a partir do momento que é instanciado na memória. Deixando de existir quando o programa é encerrado.

* **Problema 14**

O Objetivo do operador new é criar um novo estado na memória para aquela classe.

* **Problema 15**

Um construtor é um "método" utilizado para inicializar os atributos da classe. Seu nome deve ser o mesmo da classe. Ex:

public class Pessoa{

public Pessoa(){}

}

...

Pessoa p = new Pessoa();

* **Problema 16**

Será o construtor padrão que o Java implementa. Ex:

public Pessoa{

}

* **Problema 17**

Pessoa p = new Pessoa();

* **Problema 18**

É a capacidade de não expor o funcionamento do objeto, ou seja, somente a própria classe precisa saber como é seu funcionamento.

Os outros objetos que vão se comunicar com ele não precisão saber como funcionam seus métodos.

* **Problema 19**

Ordenar que o método ou atributo seja acessado por qualquer outro objeto ou de forma direta.

* **Problema 20**

Ordenar que o método ou atributo seja acessado somente pela própria classe.

* **Problema 21**

O encapsulamento em uma classe que ira Mostrar de que forma será possível acessar um atributo. Se um atributo for privado, só será possível acessar utilizando os métodos da própria classe, caso seja público poderá ser acessada de forma direta.

public class Aluno{

private String matricula;

public String getMatricula(){

return this.matricula;

}

public void setMatricula(String matricula){

this.matricula = matricula;

}

}

* **Problema 22**

Quando o encapsulamento torna um atributo da classe privado, para acessar o valor deste atributo utilizamos um método getter.

public class Aluno{

private String matricula;

public String getMatricula(){

return this.matricula;

}

...

}

* **Problema 23**

Quando o encapsulamento torna um atributo da classe privado, para definir o valor deste atributo utilizamos um método setter.

public class Aluno{

private String matricula;

public void setMatricula(String matricula){

this.matricula = matricula;

}

...

}

* **Problema 24**

1 - Classe, 2- Atributos, 3 - Métodos

* **Problema 25**

Definir o modificador, sendo +(public), -(private), #(protect), ~(default).

O nome do atributo (seguindo o padrão da linguagem) : e o tipo. Ex: - nome: String.

Problema 26modificador nomeDoMetodo(parametro: TipoParametro): TipoRetorno Ex: +getSaldo(): double

* **Problema 27**

Utilizando o termo << create >> antes do metodo que leva o mesmo nome da clas.

Ex: << create >>+Moto(marca:String, modelo: String, ano: int)

* **Problema 28**

| **Modificador** | **Tipo de Acesso** |
| --- | --- |
| + | (public) |
| - | (private) |
| # | (protect) |
| ~ | (default) |

* **Problema 29**

| **Cliente** |
| --- |
| -nome: String |
| -email: String |
| -telefone: int |
|  |
| +getNome(): String |
| +setNome(nome: String) |
| +getEmail(): String |
| +setEmail(email: String) |
| +getTelefone(): int |
| +setTelefone(telefone: int) |

* **Problema 30**

public class Cliente{

private String nome;

private String email;

private int telefone;

public String getNome(){

return this.nome;

}

public void setNome(String nome){

this.nome = nome;

}

public String getEmail(){

return this.email;

}

public void setEmail(String email){

this.email = email;

}

public String getTelefone(){

return this.telefone;

}

public void setTelefone(String telefone){

this.telefone = telefone;

}

}

* **Problema 31**

public class Produto{

private String nome;

public String getNome(){

return this.nome;

}

public void setNome(String nome){

this.nome = nome;

}

}

public class Password{

private String value;

public void Password(String value){

}

public boolean isEqual(Password p){

}

}

public class Animal{

private boolean alive;

public boolean isAlive(){

}

private void die(){

}

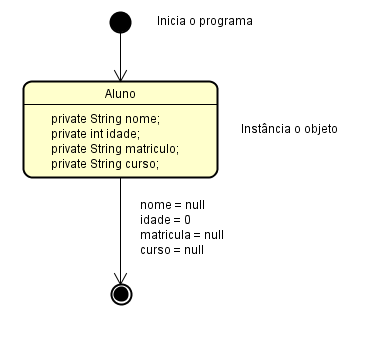
}

* **Problema 32**

| **Livro** | **ContaCorrente** | **Par** | **Impressora** |
| --- | --- | --- | --- |
| -nome: String | -saldo: double | +chave: String |  |
| -telefone: int |  | +valor: String |  |
|  |  |  |  |
| +getNome(): String | +sacar(valor: double): double |  | +imprimir(documento: Documento) |
| +setNome(nome: String) | +depositar(valor: double) |  |  |
|  | -recalcularSaldo() |  |  |

* **Problema 33**

O estado de um objeto é os valores que seus atributos possuem no momento.

[](https://github.com/carlosmondo/POO-BES2019/blob/master/1BIM/Lista-1-POO/pic/Diagrama_de_Estado_Classe_Aluno.PNG)

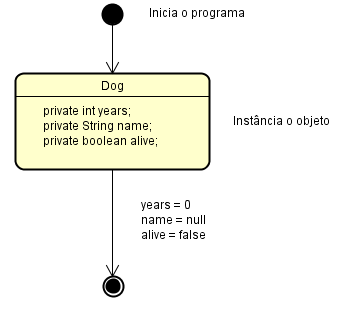
* **Problema 34** [](https://github.com/carlosmondo/POO-BES2019/blob/master/1BIM/Lista-1-POO/pic/Diagrama_de_Estado_Classe_Dog.PNG)

diagram de objetos

public class Dog(){

int = 0

String = null

boolean = false

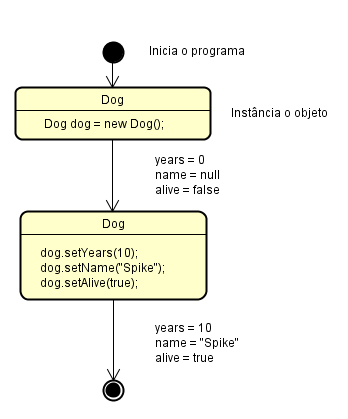
**Problema 35** [](https://github.com/carlosmondo/POO-BES2019/blob/master/1BIM/Lista-1-POO/pic/Diagrama_de_Estado_Classe_Dog_2.PNG)

diagram de objetos

public class Dog(){

Years int = 10

name String = "Spike"

alive boolean = true

* **Problema 36**

diagram de objetos

public class pug(){

idade int = 2

nome String ="Spoke"

vivo boolean = true

public class buldog(){

idade int = 0

nome String ="Spike"

vivo boolean = false

* **Problema 37**

No código logo abaixo, existem alguns problemas, como:

* + O atributo weight estar como public, tornando-se os métodos getWeight e setWeight inuteis, permitindo o acesso direto ao atributo.
  + No método main, após instânciar o objeto, o atributo name é acessado de forma direta, o que não é possivel, pois seu modificador é privado.

public class Cat{

private String name;

public double weight;

// Getters e Setters suprimidos

}

public class Main{

public static void main(String args[]){

Cat c = new Cat();

c.weight = 3.5;

c.name = "BlackCat";

}

}

**As possiveis correções seriam:**

utilizar os setters para modificar o estado dos atributos do objeto.

Tornar os atributos todos privados;

public class Cat{

private String name;

private double weight;

// Getters e Setters suprimidos

}

public class Main{

public statis void main(String args[]){

Cat c = new Cat();

c.setWeight(3.5);

c.setName("BlackCat");

}

}