

STEP JAVASCRIPT - homework 013

(ao final deste doc, uma cópia de código rascunho que foi usado em sala de aula, para ser usado como referência)

Exercicio 1 de 3:

Baseando-se nas várias maneiras de definir, declarar e modificar objetos e *métodos* que vimos na última aula, tente adaptar para javascript o seguinte código de uma aula de java:

(nota: tente obter exatamente o mesmo comportamento que havia no java, incluindo o encapsulamento)

```
public class Principal {

    public static void main(String[] args) {

        Lanterna lanterna = new Lanterna();
        Pilha pilha1 = new Pilha();
        Pilha pilha2 = new Pilha();

        lanterna.trocar(pilha1, pilha2);
        if (lanterna.ligar())
            System.out.println("Liguei a lanterna");
        else
            System.out.println("A lanterna não pode ser ligada");
    }
}

class Lanterna {

    private boolean pronta;

    boolean ligar() {
        if (pronta) {
            return true;
        } else {
            return false;
        }
    }

    void trocar(Pilha p1, Pilha p2) {
        if (p1.obtemStatus() &&
            p2.obtemStatus())
            pronta = true;
    }
}
```

```

    }
}

class Pilha {

    private float voltagem;

    boolean obtemStatus() {
        if (voltagem > 0.0f) {
            return true;
        } else
            return false;
    }
}

```

Exercício 2:

Baseando-se nas várias maneiras de definir, declarar e modificar objetos e *métodos* que vimos na última aula, tente adaptar para javascript o seguinte código de uma aula de java:

(nota: tente obter exatamente o mesmo comportamento que havia no java, incluindo o encapsulamento)

```

public class Principal {
    public static void main(String args[]) {

        // utilizando construtores
        Carro c1 = new Carro(
            new Pneu(true),
            new Pneu(true),
            new Pneu(true),
            new Pneu(true));    // teste todos cheios
        c1.verificaPneus();

        Carro c2 = new Carro(
            new Pneu(true),
            new Pneu(false),
            new Pneu(true),
            new Pneu(true));    // teste só um furado
        c2.verificaPneus();

        Carro c3 = new Carro(
            new Pneu(false),
            new Pneu(false),
            new Pneu(false),

```

```

        new Pneu(true)); // teste todos vazios
    c3.verificaPneus();
}

class Carro {
    private Pneu dianteiroDireita;
    private Pneu dianteiroEsquerda;
    private Pneu traseiroDireita;
    private Pneu traseiroEsquerda;

    // construtor
    Carro(Pneu p1, Pneu p2, Pneu p3, Pneu p4) {

//        Pneu[] pneu = new Pneu[4];
//        for (int i = 0; i < pneu.length; i++) {
//            pneu[i] = new Pneu(true);
//        }

        dianteiroDireita = p1;
        dianteiroEsquerda = p2;
        traseiroDireita = p3;
        traseiroEsquerda = p4;
    }

    void verificaPneus() {
        if (!dianteiroDireita.estaCheio())
            mensagem("Dianteiro Direito");
        if (!dianteiroEsquerda.estaCheio())
            mensagem("Dianteiro Esquerdo");
        if (!traseiroDireita.estaCheio())
            mensagem("Traseiro Direito");
        if (!traseiroEsquerda.estaCheio())
            mensagem("Dianteiro Direito");

        private void mensagem(String nomePneu) {
            System.out.println("O pneu " + nomePneu + "está furado (ou vazio).");
        }
    }
}

class Pneu {
    private boolean cheio;
    private float pressao;

    // construtor
    Pneu(boolean c) {
        cheio = c;
    }
}

```

```

    // resta o problema deste método não ser private
    // mas como está sendo usado somente pelo carro,
    // não "fere" o enunciado
    boolean estaCheio() {
        if (cheio)
            return true;
        else return false;
    }
}

```

Exercicio 3:

Represente no seu código um gato, um cão, uma andorinha e um cobra, usando os princípios da orientação a objetos que vimos até agora.

A título de observação inicial, considere os seguintes grupos de características / comportamento comum:

Cão e gato correm, andorinha voa, cobra “esgueira-se”
 Cão e gato tem quatro patas, andorinha duas, cobra nenhuma
 Cão, gato, andorinha e cobra emitem algum tipo de som
 Cão e gato e cobra tem boca, andorinha tem bico.
 Cão e gato tem pelos, andorinha penas, cobra escamas

Em seguida, crie vários objetos que representem as características acima, e chame métodos desses objetos (se houverem) que cumpra o comportamento sugerido, e exiba informações sobre essas características.

Requerido: continue se exercitando no sentido de todo o seu código estar dentro de functions. Continue tentando usar somente o **let**

=====

Entregue a pasta com o arquivo html dentro

=====

```

<script>
    function Ponto(x, y) {
        this.x = x
        this.y = y
    }

```

```

}

function Reta(p1, p2) {
    this.p1 = p1
    this.p2 = p2
}

function Triangulo(r1, r2, r3) {
    this.r1 = r1;
    this.r2 = r2;
    this.r3 = r3;
    this.listaDePontos = listaDePontos;
    let cor;
    //listaDePontos = function () {
    //    return "reta1, x" + r1.p1.x
    //}
    function listaDePontos() {
        return "reta1, x" + r1.p1.x + " e a cor " + cor
    }
}

reta2 = new Reta(
    new Ponto(10,10),
    new Ponto(20,12)
)

reta4 = reta3 = reta2
t = new Triangulo(reta4, reta3, reta2)
t2 = {
    r1 : new Reta(new Ponto(2,3), new Ponto(2,5)),
    r2 : new Reta(new Ponto(2,3), new Ponto(2,5)),
    r3 : new Reta(new Ponto(2,3), new Ponto(2,5)),
    l : function listaDePontos() {
        return "reta1 do t2, x" + this.r1.p1.x
    }
}

console.log(t2.l())

```

```
lTemp = t.listaDePontos();

p1 = new Ponto(10,10)
p2 = new Ponto(5,3)
reta = new Reta(p1, p2)

p1 = {}
p2 = {}
p1.voar = function voar() {}
p2.andar = function andar() {}

p1 = {
  voar : function voar() {}
}

p2 = {
  andar : function andar() {}
}

function Passaro(asa) {
  var asas = "xxxx";
  this.voar = voar;
  function voar() {
    console.log(asas)
  }
}

function Passaro(pata) {
  var patas = "xxxx";
  this.andar = andar;
  function andar() {
    console.log(patas)
  }
}

</script>
```