STEP Javascript homework 0180

(Ao final do doc segue o código usado em sala de aula para referência)

Exercício:

Segue abaixo um código escrito em Java, que maneja uma hierarquia de eletrodomésticos.

1. Veja se você consegue traduzir para javascript.
2. No código, existe já uma TV e um rádio. Crie outros tipos de objetos sem mudar (ou mudando o mínimo possível) a estrutura da hierarquia existente. Esses objetos devem entretanto estar necessariamente ligados à hierarquia de eletrodomésticos.
3. Crie *dois ou mais* objetos de cada tipo (duas TVs, dois rádios, etc).
4. ***Exiba, para cada objeto criado, uma lista com os nomes dos seus atributos*.** Adicionalmente, se conseguir, liste o valor do atributo junto o nome do atributo

    Estrutura hipotética a usar (imagine algo parecido):

        TV Samsung

            Nome: samsung
            Modelo: AK35
            Peso: 6,8
            (...)

    DICA e Requerido: use a forma “in” de um comando *for*

Requerido: use herança moderna (*class* e *extends. Também, se necessário, constructors*)
Requerido: o manejo dos estados dos objetos (p.ex. ligar ou desligar) deve ser controlado *pela interface do usuário* (ou seja, imagine input boxes, botões e quaisquer tags html que lhe auxiliem - *tags simples*, para que o “peso” das críticas ao dado recairam inteiramente no javascript)
Requerido: na criação e na mudança de estado de qualquer objeto, uma info relacionada deve ser exibida na página: *use a console apenas para seus próprios debugs)*. Tipo: “uma TV foi ligada”, “uma TV foi desligada”, “volume do rádio aumentado”, etc.

```
public class Principal {

	public static void main(String[] args) {
		TV tv1 = new TV(52, 110);
		Radio radio1 = new Radio(220);
		tv1.ligar();
		radio1.ligar();
		radio1.desligar();
		System.out.print("Neste momento a TV está ");
		System.out.println(tv1.isLigado() ? "ligada" : "desligada");
		System.out.print("e o Rádio está ");
```

```
            System.out.println(radio1.isLigado() ? "ligado." : "desligado.");
      }
}

abstract class Eletrodomestico {

      private boolean ligado;
      private int voltagem;

      abstract void ligar();
      abstract void desligar();

      Eletrodomestico(boolean ligado, int voltagem) {
            this.ligado = ligado;
            this.voltagem = voltagem;
      }
      public void setVoltagem(int voltagem) {
            this.voltagem = voltagem;
      }
      public int getVoltagem() {
            return this.voltagem;
      }
      public void setLigado(boolean ligado) {
            this.ligado = ligado;
      }
      public boolean isLigado() {
            return ligado;
      }
}

class Radio extends Eletrodomestico {
      public static final short AM = 1;
      public static final short FM = 2;
      private int banda;
      private float sintonia;
      private int volume;
      public Radio(int voltagem) {
            super(false, voltagem);
            setBanda(Radio.AM);
            setSintonia(0);
            setVolume(0);
      }
      public void desligar() {
            super.setLigado(false);
            setSintonia(0);
            setVolume(0);
      }
      public void ligar() {
```

```
            super.setLigado(true);
            setSintonia(88.1f);
            setVolume(25);
      }
      public int getBanda() {
            return banda;
      }

      public void setBanda(int banda) {
            this.banda = banda;
      }

      public float getSintonia() {
            return sintonia;
      }

      public void setSintonia(float sintonia) {
            this.sintonia = sintonia;
      }

      public int getVolume() {
            return volume;
      }

      public void setVolume(int volume) {
            this.volume = volume;
      }

}

class TV extends Eletrodomestico {
      private int tamanho;
      private int canal;
      private int volume;

      public TV(int tamanho, int voltagem) {
            super(false, voltagem);
            this.tamanho = tamanho;
            this.canal = 0;
            this.volume = 0;
      }
      public void desligar() {
            super.setLigado(false);
            setCanal(0);
            setVolume(0);
      }
      public void ligar() {
            super.setLigado(true);
```

```
            setCanal(3);
            setVolume(25);
        }


        public int getTamanho() {
            return tamanho;
        }

        public void setTamanho(int tamanho) {
            this.tamanho = tamanho;
        }

        public int getCanal() {
            return canal;
        }

        public void setCanal(int canal) {
            this.canal = canal;
        }

        public int getVolume() {
            return volume;
        }

        public void setVolume(int volume) {
            this.volume = volume;
        }

}
```

CÓDIGO DESENVOLVIDO EM SALA DE AULA (arquivo .js apenas)

```
// Herança e Orientação a Objeto em geral
// continuação

class Conta {
    numConta = 123
    nome = "dudu"
    saldo = 1500
}

class ContaCorente extends Conta {
    taxa = 1.2
```

```
    descontaTaxa() {
        saldo -= taxa
    }
}

// se inspecionarmos cg e c na console, veremos
// os atributos herdados, incluindo os valores
var cg = new Conta();
var c = new ContaCorente();
// nota: criamos conta apenas para compararmos.
// pode ser que em uma situação mais real,
// somente conta corrente seria criada

// inserimos ambas as contas em uma lista
var listaContas = []
listaContas.push(cg)
listaContas.push(c)

// cpmo é possível retornar a estrutura do objeto
// como um array, é possivel também percorre-lo
// na forma de uma ITERAÇÃO
for(var i = 0; i< listaContas.length; i++) {
    for(p in listaContas[i]) {
        console.log(p)
    }
}
// quando isto é usado co CLASSES, só retorna atributos,
// e não, métodos. Verifique abaixo: agora estamos
// usando o modelo de prototype para variar
// o teste

function ContaPoupanca () {
    function () {
        this.valorRemuneracao = 1.3
    }
}
```

```
cp = new ContaPoupanca()
ContaPoupanca.prototype = Object.create(Conta.prototype)
// inspecionando conta poupança, verificamos que
// o prototype não foi herdado porque ele
// não existe em Conta (que foi feita usando class)
// e não a partir de uma function construtora
```