

Galeria da Fama & Vergonha T9.2

Profa Andréa Schwertner Charão

Aluno Alberto Francisco Kummer Neto

DLSC/CT/UFSM

Dados gerais

- Observações baseadas em 17 trabalhos
- Apresenta-se a contagem de trabalhos que se enquadra em cada uma das observações

Galeria da Vergonha I: Taxa

■ Constantes mágicas

(15)

```
if(tipo == 'c')  
    return 3*horas;  
else  
    return 1.5f*horas;
```

```
public class Veiculo {  
    ...  
    public float GetCustoHora(){  
        if( tipo == 0 ) return 3;  
        else return (float) 1.50;  
    }  
}
```

```
protected double getVehicleHoursTax()  
{  
    if (this.vehicle instanceof VehicleCar) {  
        return 3.0;  
    } else return 1.5;  
}
```

Galeria da Fama I: Taxa

■ Representação com variáveis estáticas

Esqueceu o `static`!
Cada instância de
`Estacionamento` terá
uma nova cópia dos
atributos.

(2)

```
public class Estacionamento {  
    private final double precoHoraCarro = 3;  
    private final double precoHoraMoto = 1.5;
```

```
public class Estacionamento {  
»    private static final double porHoraCarros = 3.00;  
»    private static final double porHoraMotos = 1.50;
```

Galeria da Vergonha II: Estr. Simulação

- Cálculos estranhos e/ou fixos
- Código divergente
- Repetição de código

???

(13)

A JVM costuma remover laços de repetição que não fazem mudança alguma nos dados da aplicação.

```
x.Valor(System.currentTimeMillis()+(27777800*3));
```

```
for(i=0; i<=999; i++){  
    System.out.println("dando um tempo");  
}
```

```
...  
if(v1.Tipo() == 1){  
    horasaida = System.currentTimeMillis() + 1800000;  
    valor = 0.05 * ((horasaida - v1.Hora()) / 60000);  
    System.out.printf("Valor(Carro, Placa: %s): R$%.2f\n", v1.Placa(), valor);  
}  
if(v1.Tipo() == 2){  
    horasaida = System.currentTimeMillis() + 10800000;  
    valor = 0.025 * ((horasaida - v1.Hora()) / 60000);  
    System.out.printf("Valor(Moto, Placa: %s): R$%.2f\n", v1.Placa(), valor);  
}
```

Código duplicado.

Galeria da Fama II: Estr. de Simulação

- Código fácil de ler
- Tratamento de situações diferentes
- Abordagem de simulação interessante

Fácil de ler.

(4)

Trata pernoite.

```
int hours = (int) ((milli / (1000*60*60)) % 24);  
int minutes = (int) ((milli / (1000*60)) % 60);
```

```
if(entrada > saida)  
» return ((24 - entrada) + saida)*porHoraCarros;  
else  
» return (saida - entrada)*porHoraCarros;
```

```
Integer time = Math.round((System.currentTimeMillis() -  
    this.vehicleEntranceHour) / 3600000 + rand.nextInt(5));
```

Usa random.

Galeria da Vergonha III: Estruturação de Código

- Tudo num arquivo só

(3 (2+1))

```
class Veiculo{
    private final char tipo;
    private final String placa;
    public Veiculo(char xTipo, String xPlaca){
    public char getTipo(){
    }
}
class Estacionamento{
    private Veiculo veiculoEstacionado;
    private long horaEntrada;
    public Estacionamento(){
    public void addVeiculo(Veiculo xVeiculoEstacionado){
    public float rmvVeiculo(){
    }
}
public class Parte2 {
    public static void main(String[] args) {
        Veiculo v1 = new Veiculo('c', "ABC1234");
        Veiculo v2 = new Veiculo('m', "DFG5678");
        Estacionamento e1 = new Estacionamento();
        System.out.println("Para simplificar, nesse exemplo, o preco eh por segundo e nao por horas.");
        e1.addVeiculo(v1);
        try {
            Logger.getLogger(Parte2.class.getName()).log(Level.SEVERE, null, ex);
        }
        System.out.printf("v1: Preco: %.2f\n", e1.rmvVeiculo());
        e1.addVeiculo(v2);
        try {
            Logger.getLogger(Parte2.class.getName()).log(Level.SEVERE, null, ex);
        }
        System.out.printf("v2: Preco: %.2f\n", e1.rmvVeiculo());
    }
}
```

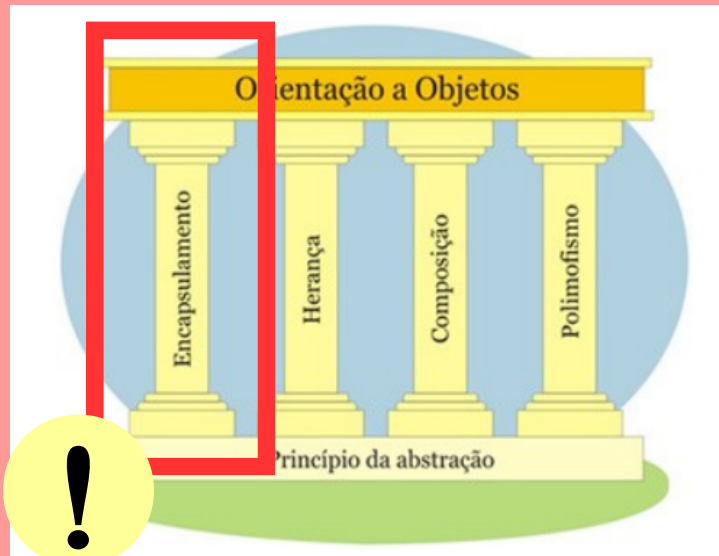
...

Galeria da Vergonha IV: Estruturação de Código

- Visibilidade pública para atributos do objeto

Isso vai contra os princípios da POO!

(2)



```
public class Veiculo {  
    public String tipo;  
    public String placa;  
}
```

```
public class Estacionamento {  
    public long datae;  
    public long datas;  
    public float valor;  
    //public long entrada;  
    //public long saida;  
    public Veiculo carro;  
}
```


Galeria da Fama III: Estruturação de Código

- Abordagem genérica para o tratamento dos veículos

(1)

Uso de
design patterns!

```
class VehicleFactory {  
    public VehicleCar car(String plate) {  
  
    public VehicleMotorcycle motorcycle(String plate) {  
  
    }  
}
```

Uso de herança para
tratar dos tipos
de veículos: note o
abstract na declaração
da classe...

```
abstract class VehicleAbstract {  
    protected String plate;  
  
    public String getPlate() {  
  
    public Void setPlate(String plate) {  
  
    }  
}
```

... mas faltou o método
abstract float getTax().

Visão geral

- Constantes mágicas
- Código difícil de compreender
- **Falta de comentários nos fontes**
- Ao desenvolver uma aplicação com POO, sempre use relações que **minimizem o acoplamento**
- Coisas imutáveis em *runtime* marcadas como **static final**
- Código expressivo
- *Design patterns* podem auxiliar no desenvolvimento de aplicações complexas
- Uso de IDE