Lab 04 - Sistema de Apostas

**Universidade Federal de Campina Grande – UFCG**
**Centro de Engenharia Elétrica e Informática – CEEI**
**Unidade Acadêmica de Sistemas e Computação – UASC**

**Disciplina: Laboratório de Programação 2**

**Laboratório 04**

**Como usar esse guia:**

- Leia atentamente cada etapa
- Referências bibliográficas incluem:
  - material de referência desenvolvido por professores de p2/lp2 em semestres anteriores (ONLINE)
  - o livro Use a cabeça, Java (LIVRO-UseCabecaJava)
  - o livro Java para Iniciantes (Livro-JavaIniciantes)
- Quadros com dicas tem leitura opcional, use-os conforme achar necessário
- Preste atenção nos trechos marcados como importante (ou com uma exclamação)!

**Sumário**

**Introdução 1**

**Sistema de Apostas 2**

1. Inicializar Sistema 2
2. Cadastrar e Listar Cenários 3
3. Cadastrar e Listar Apostas 3
4. Fechar Apostas (Concretizar Cenário) 4

**... Outros casos de uso 4**

# Introdução

Neste laboratório começaremos a trabalhar através de uma especificação com casos de uso seguindo o modelo que será usado no projeto. Uma especificação determina os requisitos que um usuário tem para o software. Cada funcionalidade do sistema a ser implementado é descrita em um caso de uso, ou seja, um cenário que descreve como essa funcionalidade é usada no sistema.

**Ao contrário dos outros laboratórios, você não deve criar a classe main/menu**. Neste laboratório você está construindo a base (*backend*) do sistema. Sua obrigação é oferecer classes com métodos bem definidos para que qualquer desenvolvedor

possa pegar o que você desenvolveu para implementar a interface gráfica/texual/web com o usuário (*frontend*).

Para facilitar a vida do desenvolvedor de *frontend*, seu sistema deve ter uma classe Facade contendo todos os métodos e operações que o desenvolvedor responsável por implementar a interface gráfica ou textual precisa. Exemplo de uma classe Main que poderia ser implementada utilizando a classe Facade que seu sistema irá oferecer:

```
public class Main {
    public static void main(String[] args) {
        Facade facade = new Facade();
         int centavos = 100000;
         double taxa = 0.01;
        facade.inicializa(centavos, taxa);
         facade.criaCenario("A maioria irá tirar mais do que 7 na prova!");
        // ... etc
    }
}
```

**Dica de implementação!**

Facade = Fachada. Uma classe que abstrai todo um sistema (ou um conjunto de subsistemas) para facilitar o uso dessas partes abstraídas.

A estratégia de criar uma única classe de entrada (como a classe Facade) é bastante comum e facilita a vida do desenvolvedor. Uma classe como a classe Facade pode compor um ou mais controladores do sistema. Exemplo:

```
public class Facade {
    // ...
    public void salva() {
        apostasController.salva();
         cenariosController.salva();
    }
    // ...
}
```

Idealmente a Facade deve trabalhar apenas com os tipos mais básicos do sistema. Isso diminui o acoplamento do sistema. Dessa forma, faça métodos que usem apenas os tipos básicos nos parâmetros e no retorno (Strings, int, double, etc.).

Uma fachada não é o mesmo que um controlador. Uma facade deve oferecer apenas métodos simples e de uso geral do usuário. Um controlador pode ter mais métodos e é responsável por fazer tratamento da lógica de negócio. Em geral a facade apenas delega ações para um ou mais controladores que o compõe.

# Sistema de Apostas

No projeto atual você deve implementar um sistema de apostas que possui um caixa inicial. Neste programa, são cadastrados cenários que podem acontecer, ou não. Por exemplo, pode-se imaginar o cenário em que "mais da metade da turma irá pagar física por média". Para cada cenário, é possível fazer apostas com o nome do

apostador, quantia de aposta (R$) e uma previsão (o cenário vai acontecer ou o cenário não vai acontecer).

Em determinado momento, o sistema concretiza o cenário (dizendo se ocorreu ou não) tornando as apostas vencedoras e perdedoras (se elas acertaram ou não a previsão que fizeram). Este sistema contará ainda com: i) apostas especiais, em que um prêmio especial é oferecido pelo próprio sistema de apostas, e; ii) por apostas seguradas, que protege o apostador em determinadas situações. Além disso o sistema deve imprimir os cenários ordenados por diferentes critérios.

Para cada aposta perdedora, o sistema retira uma parte da quantia apostada para o seu próprio caixa (o resto é distribuído para os vencedores da aposta). O sistema também irá retirar dinheiro desse caixa para colocar nas apostas prêmios (quando essas forem cadastradas). É importante sempre ter controle desse valor em caixa.

## 1. **Inicializar** Sistema

Durante a inicialização do sistema, deve-se receber um valor indicando a quantidade atual de dinheiro (em centavos) que o sistema possui e a porcentagem que deve ser retirada de cada aposta perdedora para ir ao caixa do sistema (o resto do valor é usado para premiar as apostas vencedoras).

O valor de porcentagem não será alterado ao longo do programa. Deve ser possível conseguir recuperar o caixa atual do sistema em qualquer momento.

A Facade deve oferecer métodos para:
- Inicializar o sistema com um valor e taxa (exemplo: 100000 e 0.01)
    - void inicializa(int caixa, double taxa)
- Recuperar o valor em caixa do sistema (em centavos).
    - int getCaixa()

## 2. Cadastrar e Listar Cenários

Cada cenário apresenta uma descrição de uma possível situação futura que poderá (ou não) acontecer. Os cenários iniciam como não finalizados e passam a ser ditos como finalizados quando são encerrados.

Ao encerrar um cenário, deve ser informado ao sistema se o mesmo ocorreu ou não. Todo cenário é identificado por uma numeração inteira. O primeiro cenário recebe a numeração 1, o segundo cenário criado recebe a numeração 2 e assim sucessivamente.

Cada cenário tem uma representação textual no formato “NUMERAÇÃO - DESCRIÇÃO - ESTADO”, onde ESTADO pode ser “Finalizado (ocorreu)”, “Finalizado (n ocorreu)” ou “Não finalizado”. Exemplos:

“1 - A maioria irá tirar mais do que 7 na prova! - Não finalizado”
“2 - O professor irá para a aula sobre GRASP com um café! - Finalizado (ocorreu)”

A Facade deve ter métodos para:
- Cadastrar cenários e retornar a numeração do cenário cadastrado
    - int cadastrarCenario(String descricao)
- Retornar a representação textual de um cenário (a partir da numeração)

- String exibirCenario(int cenario)

- Retornar a representação textual de todos os cenários cadastrados no sistema (um por linha, em ordem de cadastro)
  - String exibirCenarios()

## 3. Cadastrar e Listar Apostas

É possível fazer apostas para diferentes cenários. Todas as apostas são diferentes entre si e tem como atributos: nome do apostador, valor da aposta e o previsão (VAI ACONTECER ou N VAI ACONTECER).

No cenário de exemplo (“A maioria irá tirar mais do que 7 na prova!”) é possível cadastrar uma aposta feita por Matheus, valendo 10000 centavos concordando com o cenário. É importante observar que é possível ter duas apostas diferentes com o mesmo nome do apostador, valor e tipo. Um mesmo apostador pode fazer diferentes apostas, inclusive de tipos diferentes.

Cada aposta é representada textualmente pelo formato “NOME - VALOR - PREVISÃO”. Seria um exemplo de representação:

“Matheus Gaudencio - R$100,00 - VAI ACONTECER”

A Facade deve ter métodos para:

- Cadastrar a aposta com nome do apostador, quantia apostada e previsão em um cenário através do número do cenário.
  - void cadastrarAposta(int cenario, String apostador, int valor, String previsao)
- Retornar o valor total das apostas feitas em um cenário.
  - int valorTotalDeApostas(int cenario)
- Retornar o número de apostas feitas em um cenário.
  - int totalDeApostas(int cenario)
- Gerar a representação textual das apostas de um cenário (uma aposta por linha).
  - String exibeApostas(int cenario)

## 4. Fechar Apostas (Concretizar Cenário)

Em determinado momento o sistema deve encerrar as apostas e fechar o cenário. O sistema deve ter uma operação em que finaliza o cenário indicando se o mesmo ocorreu, ou não. Quando um cenário acontece (cenário ocorreu), as apostas que previram que o cenário “VAI ACONTECER” são consideradas vencedoras e as demais perdedoras. Quando o cenário não acontece (não ocorreu) as apostas que previram que o cenário “N VAI ACONTECER” são consideradas vencedoras e as demais perdedoras.

Quando o sistema concretiza o cenário, o valor de todas as apostas perdedoras é somado. Desse valor, parte é retirado (de acordo com a taxa definida na inicialização do sistema) e acrescentado ao caixa do sistema. Esse valor deve ser sempre inteiro e arredondado para baixo. O resto do dinheiro será usado para ser  rateado igualmente entre os apostadores vencedores.

Considere o exemplo de 6 apostas do cenário discutido anteriormente:

- “Francisco Cisco - R$200,00 - N VAI ACONTECER”
- “Anonimo - R$1,99 - N VAI ACONTECER”
- “Matheus Gaudencio - R$100,00 - VAI ACONTECER”
- “Livia Maria - R$300,00 - VAI ACONTECER”
- “Raquel Lopes- R$200,00 - VAI ACONTECER”
- “Matheus Gaudencio - R$100,00 - VAI ACONTECER”

Se o cenário se concretizar, as apostas de Matheus, Lívia e Raquel são vencedoras. As apostas de Francisco e Anonimo são perdedoras. O sistema então soma R$200,00 com R1,99 e multiplica pela taxa inicial definida no sistema (por exemplo, 0.01). Isto dá um total de R$2,0199, ou 201 centavos (arredondando para baixo). O sistema acrescenta então 201 centavos ao seu caixa e marca o cenário como encerrado. O sistema deve armazenar também qual o bolo a ser rateado entre os vencedores. No exemplo acima, R$ 199,98 devem ser distribuídos entre os vencedores.

A Facade deve ter métodos para:

- Encerrar um cenário (indicando se ocorreu ou não).
    - void fecharAposta(int cenario, boolean ocorreu)
- Retornar o valor total de um cenário encerrado que será destinado ao caixa
    - int getCaixaCenario(int cenario)
- Retornar o valor total de um cenário encerrado que será destinado a distribuição entre as apostas vencedoras
    - int getTotalRateioCenario(int cenario)

## ... Outros casos de uso

Nas próxima semana publicaremos mais dois casos de uso neste documento.

# Entrega

Faça o sistema de apostas que tenham as funcionalidades descritas até o momento da entrega. É importante que o código esteja devidamente documentado. Junto a cada entrega, **você deve entregar também um design de classes do sistema**.

Ainda, você deve entregar um programa com testes para as classes com lógica testável. Tente fazer um programa que possa ser devidamente testado e que explore as condições limite.

Este laboratório terá múltiplas entregas. Para cada entrega você deve entrar o que está descrito abaixo:

- 10/7 (segunda): Sistema completo com os casos de uso 1-4
- 17/7 (segunda): Sistema completo com os casos de uso 1-6
- 26/7 (quarta): Sistema completo com os casos de uso 1-7

Para a entrega do design (e para os demais designs que forem feitos) você deve prover uma imagem (png/jpg) no formato LAB4_NOME.PNG.

Faça um **zip** da pasta do seu projeto. Coloque o nome do projeto para: LAB4_PARTE1_SEU_NOME e o nome do zip para LAB4_PARTE1_SEU_NOME.ZIP. Exemplo de projeto: LAB4_PARTE1_MATHEUS_GAUDENCIO.ZIP. Este zip deve ser submetido pelo Canvas.

Seu programa será avaliado pela corretude e, principalmente, pelo DESIGN do sistema. É importante:

- Usar nomes adequados de variáveis, classes, métodos e parâmetros.
- Fazer um design simples, legível e que funciona. É importante saber, apenas olhando o nome das classes e o nome dos métodos existentes, identificar quem faz o que no código.