Lab 04 - Sistema de Apostas - Casos de Uso parte 2

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

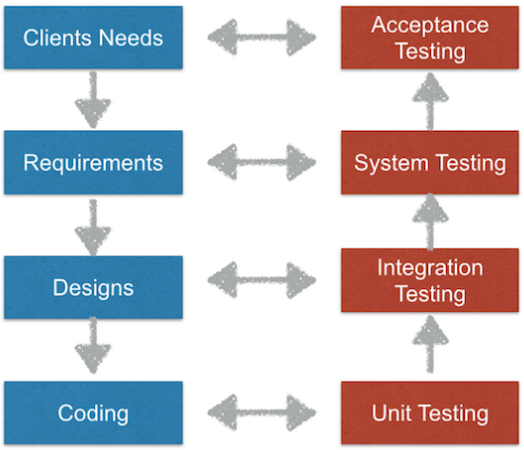
Disciplina: Laboratório de Programação 2

Laboratório 04 - Parte 2

Primeiro... uma pausa para testes de aceitação!

A atividade de teste busca VALIDAR se a especificação do programa é o que o usuário realmente deseja e VERIFICAR se o que foi desenvolvido segue essa especificação. Existem assim diferentes pontos do programa a serem testados e diferentes testes com diferentes tipos de funções.

- **Testes de unidade** são feitos para testar a menor unidade de código que existe (em geral, classes em códigos OO).
- **Testes de integração** são desenvolvidos para testar a integração de duas partes distintas de código. Por exemplo: um componente de banco de dados e o de lógica do sistema.
- **Testes de sistema** buscam verificar o sistema como um todo e, em geral, exploram aspectos como desempenho, segurança e o funcionamento geral do sistema.
- **Testes de aceitação** validam se o que o programa faz é o que o usuário realmente deseja do sistema.



(Foto original de https://usersnap.com/blog/types-user-acceptance-tests-frameworks/)

Neste laboratório iremos primeiramente usar testes de aceitação para os primeiros casos de uso (1 a 4). Os testes de aceitação estão aqui.

5. Cadastrar Cenário Bônus

Alguns cenários precisam dar incentivos aos apostadores. Os cenários que não

parecem muito óbvios ou que são puramente aleatórios (ex.: O resultado do dado será maior que três) precisam de um incentivo para que as pessoas apostem. Uma maneira de incentivar apostas em tais cenários é oferecer um bônus aos vencedores.

Seu sistema deve permitir cadastrar agora um novo tipo de cenário que ofereça bônus. Em tais cenários, o bônus é um valor que é imediatamente retirado do caixa do sistema e colocado no cenário quando o mesmo é criado. Esse valor passa a fazer parte da quantidade em dinheiro a ser distribuída entre os vencedores (ou seja, passa a ser parte do valor retornado no getTotalRateioCenario.

A representação de tais cenários será na forma: “NUMERAÇÃO - DESCRIÇÃO - ESTADO - BÔNUS”. Exemplo: “1 - O resultado do dado será maior que três - Não finalizado - R$ 100,00”

A Facade deve ter métodos para:
- Cadastrar cenários com bônus
    - int cadastrarCenario(String descricao, int bônus)

6. Cadastrar Aposta Assegurada

Um seguro é uma proteção dada ao apostador para cobrir parte de suas perdas. As apostas podem ser de três tipos:
- Aposta sem seguro
- Aposta assegurada por valor
- Aposta assegurada por taxa

Ao criar uma aposta assegurada, além dos atributos padrões da aposta, deve ser passado o tipo (VALOR ou TAXA), o próprio valor ou taxa a ser assegurado e o custo do seguro. Todos os seguros são bancados e recebidos pelo caixa do sistema. Assim, ao criar um aposta assegurada, o custo do seguro é repassado ao caixa do sistema.

Na finalização da aposta, se a mesma for decretada vencedora nada de diferente acontece. Se, no entanto, a aposta é considerada perdedora, a parte que cabe ao sistema é retirada (taxa definida na inicialização vezes o valor da aposta) e colocada em caixa. Em seguida, o valor assegurado (ou a taxa assegurada vezes o valor da aposta) é retirado do caixa. Este dinheiro será utilizado para pagar o seguro.

Num sistema de taxa 1% onde é feita uma aposta no valor de R$ 1000,00, com valor assegurado de R$ 200,00 e de custo de seguro de R$ 50,00, as seguintes ações acontecerão:
- Durante a criação da aposta, o caixa do sistema receberá R$ 50,00
- Durante o final da aposta…
    - ...se a mesma for vencedora, nada é alterado no sistema
    - …se a mesma tiver perdido:
        - R$ 10,00 são colocados no caixa (0.01 * R$ 1000,00)
        - R$ 200,00 são retirados do caixa
        - (ou seja, no final o caixa sofre um decréscimo de R$ 190,00)

Na aposta assegurada por taxa, o valor assegurado é o valor da aposta vezes a taxa assegurada. Por exemplo, em uma aposta de R$1000,00, e com seguro de 30%, o valor assegurado é de R$300,00.

Uma aposta com seguro apresenta uma representação única no formato: “NOME - VALOR - PREVISÃO - ASSEGURADA (TIPO) - VALOR”. São exemplos de apostas

asseguradas:
- “Matheus Gaudencio - R$100,00 - VAI ACONTECER - ASSEGURADA (VALOR) - R$ 200,00”
- “Matheus Gaudencio - R$100,00 - VAI ACONTECER - ASSEGURADA (TAXA) - 5%”

A qualquer momento antes da finalização de um cenário, uma aposta assegurada por valor pode virar uma aposta assegurada por taxa e vice-versa. Não há custo para realizar a conversão. Toda aposta assegurada é identificada unicamente no cenário por um número inteiro. Você é livre para gerar essa identificação.

A Facade deve ter métodos para:
- Cadastrar apostas assegurados
  - int cadastrarApostaSeguraValor(int cenario, String apostador, int valor, String previsao, int valor, int custo)
  - int cadastrarApostaSeguraTaxa(int cenario, String apostador, int valor, String previsao, double taxa, int custo)
  - int alterarSeguroValor(int cenario, int apostaAssegurada, int valor)
  - int alterarSeguroTaxa(int cenario, int apostaAssegurada, double taxa)

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.

Para a entrega, faça um **zip** da pasta do seu projeto. Coloque o nome do projeto para: LAB4_PARTE2_SEU_NOME e o nome do zip para LAB4_PARTE2_SEU_NOME.ZIP. Exemplo de projeto: LAB4_PARTE2_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.