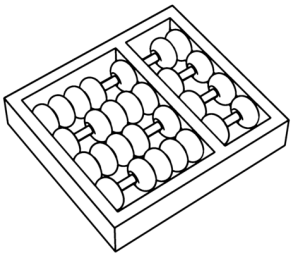

MC322A - Programação Orientada a Objetos
Instituto de Computação - Unicamp
Segundo Semestre de 2016
Profa. Esther Colombini `esther@ic.unicamp.br`
PED: Vinicius Viali `viniciusviali@gmail.com`

`http://www.ic.unicamp.br/~esther/teaching/2016s2/mc322`

Trabalho Final da Disciplina

1 Objetivo

Estudo e implementação dos conceitos dos conceitos de programação orientada a objetos abordados na disciplina na construção de uma variante do jogo banco imobiliário.

2 Tópicos Abordados

Os seguintes conceitos deverão ser adequadamente incorporados no sistema a ser desenvolvido:

- Classes, variáveis e métodos
- Visibilidade
- Herança
- Entrada e saída de dados
- Relacionamento de Associação
- Classes Abstratas
- Polimorfismo
- Interface
- Exceptions
- Arquivos

Além disso, utilizaremos diagramas de classe UML para representação do projeto de classes.

3 Regras do Jogo

As regras do jogo devem seguir as regras oficiais, como descrito em http://regras.net/como-jogar-banco-imobiliario/. Entretanto, cada equipe é livre para definir sua versão do Banco, desde que as regras sejam seguidas. As cartas de sorte/revés e os locais (tipos e valores) devem ser definidos pela equipe.

No Jogo existem:

- 1 Banco detentor inicialmente de:
  - 32 casas
  - 12 hotéis
  - 28 títulos de propriedades
  - 30 cartões SORTE/REVÉS
  - 380 notas (Verificar quantidades de cada tipo. Subtrair a quantidade entregue a cada jogador)
- 2 a 6 jogadores
- 2 dados (com valores de 1-6)
- 1 tabuleiro contendo 40 locais, sendo que um local pode ser:
  - Uma propriedade - que pode ser adquirida pelo jogador (são 28 no total), sendo que cada propriedade pode ser:
    - um terreno (com até 4 casas ou 1 hotel) que possui valor para aluguel simples, com casas, hotel e hipoteca
    - um negócio (companhia de táxi, aeroporto, etc) que possui valor básico a ser pago ao proprietário
  - Um local de utilidade (1 lucro/dividendos e 1 imposto de renda)
  - Um local de sorte/revés (6 no total)
  - Um ponto de passagem (1 prisão, 1 parada livre, 1 vá para prisão e 1 ponto de partida)

Uma partida tem de 2 a 6 jogadores e cada jogador recebe inicialmente:

- 8 notas de $1
- 10 notas de $5
- 10 notas de $10
- 10 notas de $50
- 8 notas de $100
- 2 notas de $500

Todos os jogadores partem do mesmo ponto de início. A ordem das jogadas deve ser realizada por sorteio.

Os valores obtidos pelos dados devem ser valores aleatoriamente sorteados. O mesmo acontece para a carta de sorte/revés retirada pelo jogador.

A decisão de comprar deve seguir os limites do jogo, mas cada grupo é livre para implementar o seu próprio modelo de decisão. O mesmo acontece para vendas, trocas e hipotecas.

Se um jogador falir, é preciso removê-lo da partida.

Para acompanhar o desenvolvimento do jogo, você deve, a cada jogada, armazenar em um arquivo texto (e imprimir na tela, caso deseje):

- o número da jogada

- o estado de cada local do tabuleiro (que jogadores estão nele, se tem casa, hotel, e se tem dono)
- o valor em dinheiro e as propriedade de cada jogador ainda na partida
- o saldo do banco
- se algum jogador faliu
- as operações de compra, troca, pagamento/recebimento, venda ou hipoteca realizados na partida

A Figura 1 apresenta uma possível versão do jogo a ser implementada.

Figura 1: Exemplo de tabuleiro a ser implementado.

O número de jogadores da partida deve ser indicado pelo usuário.

Ao final do projeto, responda: **Qual o número de iterações necessário para que alguém ganhe uma partida em um jogo de 2, 3, 4, 5 ou 6 jogadores?**

3.1 Grupos

Os grupos deverão ser compostos de, no máximo, 3 alunos.

3.2 Avaliação

O trabalho será avaliado segundo os seguintes critérios:

- **Corretude (25%):** O código não deve possuir warnings ou erros de compilação e não deve emitir exceptions em nenhuma situação.
- **Projeto (25%):** A qualidade do projeto realizado de acordo com os princípios de orientação a objetos. Quanto mais conceitos corretamente aplicados, maior a nota do projeto.
- **Aderência ao Enunciado (25%):** A implementação deve realizar o que é requisitado no enunciado e nas regras a serem seguidas. atentando para todos os avisos e observações.
- **Comentários (10%):** Os comentários devem ser suficientes para explicar os trechos mais importantes da implementação.
- **Convenções (15%):** A implementação deve seguir as convenções do Java. O código deve ser indentado, modularizado e hierarquizado para melhor organização.

Um diagrama UML deverá ser construído para representar o projeto do sistema proposto.

3.3 Submissão e Apresentação

A trabalho deve ser submetido por um dos membros do grupo no sistema Moodle (https://www.ggte.unicamp.br/ea/) na área correspondente à disciplina. Para isso, procure a atividade **Trabalho final** e submeta o arquivo zipado contendo todos os arquivos necessários para execução do projeto e a imagem do diagrama de classes criado. A atividade deverá ser submetida até as 23:55h do dia 12/12/2016. As apresentações devem ser agendadas pelo grupo para um horário entre os dias 12/12/2016 à 13/12/2016.