Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina de Arquitetura e Projeto de Sistemas
GABARITO – AD1 1° semestre de 2017.

Nome:
Polo:
Matrícula:

Observações:

1. Prova com consulta.

LER ATENTAMENTE AS INSTRUÇÕES A SEGUIR:

1. As respostas devem ser enviadas em um **único arquivo em formato exclusivamente .PDF, não compactado**. Além disso, o conteúdo deste arquivo deve **seguir exatamente o *template* das respostas**, caso exista. Caso não atenda a estes pontos, a **AD não será corrigida**. ADs enviadas no MODO RASCUNHO também **não serão corrigidas**. ADs MANUSCRITAS ou ESCANEADAS também **não serão corrigidas**.
2. Como a avaliação à distância é individual, caso sejam constatadas semelhanças entre provas de alunos distintos, **será atribuída a nota ZERO** a TODAS as provas envolvidas. As soluções para as questões podem ser buscadas por grupos de alunos, mas a redação final de cada prova tem que ser distinta.
3. Além disso, às questões desta AD respondidas de maneira muito semelhantes às respostas oriundas dos gabaritos já publicados de ADs e APs de períodos anteriores, **será atribuída a nota ZERO**, incluindo também cópias diretas, indiretas (semelhanças/paráfrases) ou sem sentido de tópicos dos slides das aulas. A AD é uma atividade de pesquisa (trabalho da disciplina) e deve ser elaborada como tal, não se atendo somente ao conteúdo dos slides das aulas.
4. Por fim, a pesquisa na Internet e em livros é estimulada, devendo ser referenciada na AD, mas as respostas devem ser construídas com as palavras do próprio aluno e atender diretamente ao que pede à questão, evitando respostas prolixas ou extensas. Às respostas copiadas ou semelhantes a soluções da Internet ou de livros, e/ou que não atendem (fora do escopo) ou excedem demasiadamente ao que pede a questão, **será atribuída a nota ZERO**.

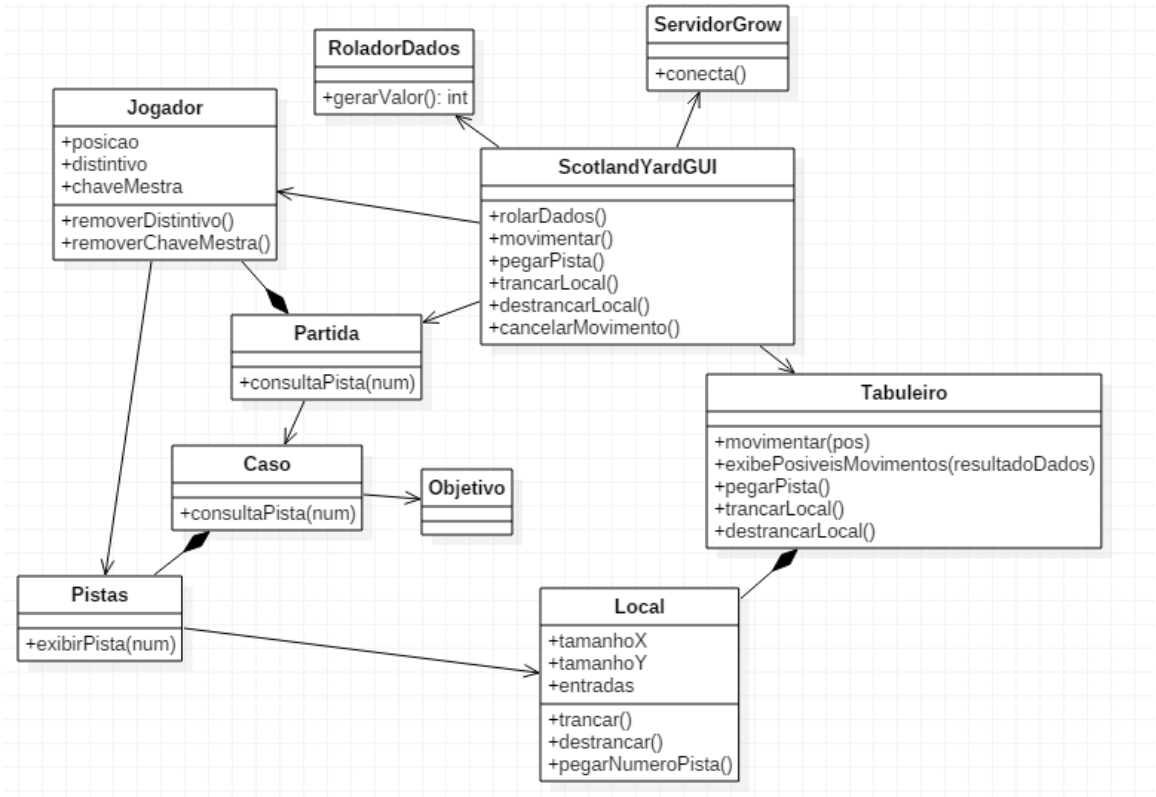
Questão 1 **[10 pontos]**

Considere a situação em que você foi contratado pela empresa brasileira Grow para gerar a documentação para viabilizar a construção do jogo **Scotland Yard** para as plataformas Windows e Android. O jogo **Scotland Yard** simula os processos verdadeiros de uma investigação criminosa nos tempos de Sherlock Holmes, possuindo diversos casos distintos onde os jogadores, seguindo as pistas, devem desvendar alguns mistérios que são pedidos pelo caso escolhido (assassino, arma, motivo, etc.).

O tabuleiro em que se passa o jogo é composto de locais onde os "detetives" leem pistas sobre o caso a desvendar. Cada jogador, na sua vez deve rolar um dado e andar no tabuleiro até chegar ao local desejado. Ao chegar, o jogo informará ao jogador qual pista que aquele local possui referente ao caso escolhido para a partida. O jogo termina quando um dos participantes desvenda o caso e vai à casa de Sherlock Holmes. Se as conclusões do jogador estiverem corretas, então ele ganha o jogo. Porém, se estiverem erradas, então esse jogador sai do jogo. O jogo continua até que alguém desvende o caso. Para manter a ambientação, o jogo não permitirá salvar a sessão ou carregar uma sessão antiga. Além disso, no inicio de cada turno, o jogador deverá clicar na tela para rolar os dados de sua movimentação para então a interface exibir as possíveis movimentações do jogador com o resultado obtido.

Para mais informações, acesse o vídeo referente às regras originais do jogo: https://www.youtube.com/watch?v=JHequuL-lcU

Dado o diagrama de classes abaixo que representa o atual esboço da **Grow** para o desenvolvimento do jogo, faça:



a) [2.0 pontos] Calcule o **grau de dependência direto** e **grau de dependência indireto** de cada uma das classes apresentadas no diagrama e exponha claramente quais são as classes relacionadas. ***A nota será atribuída caso o aluno indique as classes corretas, e não apenas o valor correto do grau***. Responda conforme o *template* da Tabela 1.

Tabela 1 – *Template* de tabela

CLASSE	GD Direto	CLASSES	GD Indireto	CLASSES

b) [0.5 pontos] Imagine uma situação em que o jogo entrou em produção. Baseado no diagrama de classes, quais classes poderiam falhar caso houvesse um defeito na classe "Jogador"? Justifique.

c) [2.0 pontos] Elabore o diagrama de casos de uso que satisfaça a descrição dada, mantendo-o coerente com o modelo de classes conceitual;

d) [1.5 pontos] Faça a descrição do caso de uso referente à movimentação do jogador no tabuleiro, que também inclui o rolamento de dados, conforme o *template* da Tabela 2;

Tabela 2 – *Template* para Descrição de Casos de Uso

Nome:	<definir o nome do caso de uso>
Objetivo:	<descrever o objetivo do caso de uso>
Atores:	<descrever os atores que interagem com o caso de uso>
Pré-condições:	<descrever as pré-condições a serem atendidas para que o caso de uso possa ser executado>
***Trigger*:**	<definir que evento dispara a execução desse caso de uso>
Fluxo Principal:	<descrever o fluxo principal do caso de uso>
Fluxo Alternativo:	<descrever os fluxos alternativos do caso de uso, indicando que evento dispara cada um deles. Cada fluxo deve ser nomeado ,<Numero do fluxo principal>.<Numero do fluxo alternativo>. Exemplo: 3.1, 3.2, 4.1
Pós-condições:	<definir que produto ou resultado concreto o ator principal obterá ao final da execução do fluxo básico>
Regras de negócio:	<listar as regras de negócios que devem ser respeitadas na execução do caso de uso. Cada regra deve ser nomeada RN1, RN2 etc., e ser referenciada em algum fluxo do caso de uso (básico ou alternativo)>

e) [4.0 pontos] Construa o Diagrama de Sequência que ilustre um turno típico do jogador, que deve incluir:

   a. Rolar dado de movimento
   b. Movimentar no tabuleiro
   c. Adquirir pista
   d. Trancar local
   e. Abrir local

Gabarito:

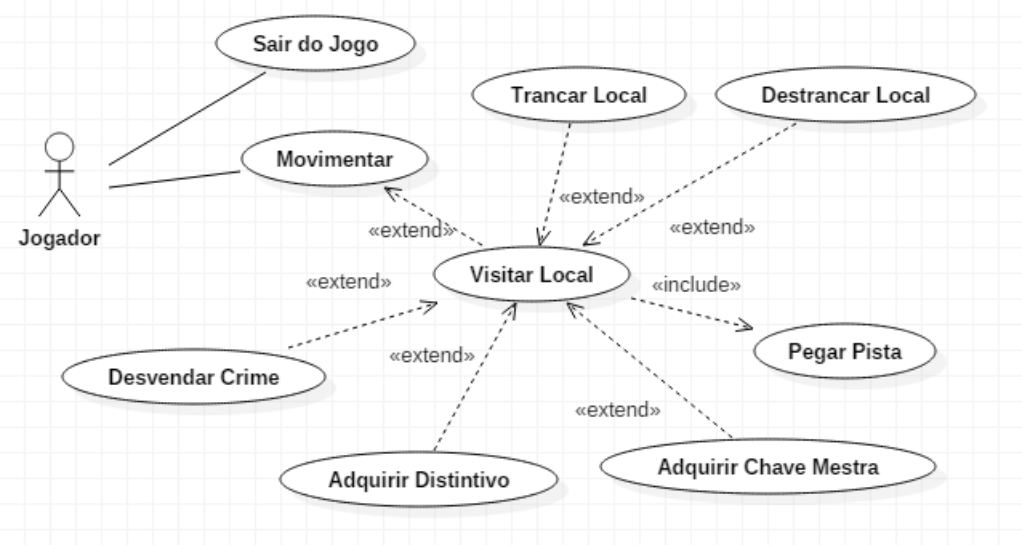
Questão 1

a) 2.0 pontos
+0.1 para cada GDD correto
+0.1 para cada GDI correto

CLASSE	GD Direto	CLASSES	GD Indireto	CLASSES
ServidorGrow	0	-	0	-
Jogador	1	Pistas	2	Pistas Local
ScotlandYardGUI	5	ServidorGrow Jogador Tabuleiro RoladorDados Partida	9	ServidorGrow Jogador Tabuleiro Local RoladorDados Partida Caso Objetivo Local
Tabuleiro	1	Local	1	Local
Local	0	-	0	-
RoladorDados	0	-	0	-
Partida	2	Jogador Caso	5	Jogador Caso Objetivo Pistas Local
Caso	2	Objetivo Pistas	3	Objetivo Pistas Local
Objetivo	0	-	0	-
Pistas	1	Local	1	Local

b) 0.5 pontos
As classes que poderiam falhar devido a um defeito na classe “Jogador” são “ScotlandYardGUI” e “Partida”, pois estas dependem diretamente de métodos ou atributos da classe “Jogador”.

c) 2.0 pontos


Sair do Jogo
Trancar Local
Destrancar Local
Movimentar
Jogador
«extend»
«extend»
«extend»
«extend»
Visitar Local
«include»
Pegar Pista
Desvendar Crime
«extend»
«extend»
Adquirir Distintivo
Adquirir Chave Mestra

d) 1.5 pontos

Nome:	Movimentação
Objetivo: [0.1 pontos]	Realizar a movimentação do turno
Atores: [0.1 pontos]	Jogador
Pré-condições: [0.1 pontos]	Estar na vez do jogador
***Trigger*:** [0.1 pontos]	Selecionar a opção para rolar os dados
Fluxo Principal: [0.5 pontos]	1. O sistema rola os dados de movimentação do jogador 2. O sistema exibe o resultado dos dados 3. O sistema exibe as possíveis movimentações dado a posição atual do jogador 4. O jogador escolhe a posição destino 5. O sistema atualiza a posição do jogador
Fluxo Alternativo: [0.2 pontos]	4.1: O usuário informa uma posição inválida. 1. O sistema exibe uma mensagem de erro 2. Retorna para o passo 3 do fluxo principal
Pós-condições: [0.2 pontos]	Movimentação concluída e jogador atualizado para a nova posição
Regras de negócio: [0.2 pontos]	Nenhuma

e) 4.0 pontos

- 0.6 pontos: Rolar dados –> Gerar Valor –> Exibir movimentos
- 0.4 pontos: Movimentar –> Movimentar(pos)
- 0.2 pontos: Opt[entrou local]
- 0.2 pontos: Opt[Trancado]
- 0.2 pontos: Alt[chave mestra] –>Caso contrario
- 0.6 pontos: DestrancarLocal –> destrancarLocal –> destrancar –> RemoverDistintivo
- 0.2 pontos: (Caso Contrario) CancelarMovimento
- 0.8 pontos: PegarPista–>PegarPista–>PegarNumeroPista –>ConsultaPista –> ConsultaPista –> ExibirPista
- 0.2 pontos: Opt[trancar]
- 0.6 pontos: Trancar –> trancar –> trancar –> Remover Chave