Instituto Federal de Minas Gerais - Campus Ouro Branco
Curso: Bacharelado em Sistemas de Informação
Disciplina: Programação para dispositivos móveis
Professor: Saulo Henrique Cabral Silva (saulo.cabral@ifmg.edu.br)

Trabalho Prático 2
Batalha de Cartas Colecionáveis

Valor: 7 pontos **Data da entrega: 05/10/2023**

Objetivos: Consiste em praticar conceitos relacionados à programação Orientada a Objetos e prática na linguagem de programação Kotlin.

Descrição:

Seu José é um apaixonado por jogos de cartas colecionáveis que é um tipo de jogo de cartas em que os jogadores constroem seus próprios baralhos personalizados a partir de uma coleção de cartas disponíveis. Cada carta representa um personagem, criatura, item ou habilidade com atributos e características únicas. Os jogadores utilizam essas cartas para competir uns contra os outros seguindo as regras estabelecidas pelo jogo.

Neste trabalho prático, Seu José pede a você caro estudante do curso Bacharelado em Sistemas de Informação que desenvolva um programa que simule um jogo de cartas colecionáveis semelhante aos conhecidos *Magic* e *Yu-Gi-Oh*. O jogo consiste em utilizar estratégias para posicionar monstros no tabuleiro, equipá-los com cartas de equipamento e realizar ataques contra o oponente. Como entrada, a solução recebe um arquivo onde Seu José anotou todas as cartas que possui. A principal diferença neste jogo, é que Seu José pretende compartilhar o seu baralho com um amigo (adversário), para se divertir e passar o tempo em boa companhia.

Durante uma rodada, os jogadores podem escolher entre as seguintes ações: a) Posicionar um novo monstro no tabuleiro; b) Equipar um monstro com uma carta de equipamento; c) Descartar uma carta da mão; d) Realizar um ataque contra o oponente; e) Alterar o estado de um monstro (ataque/defesa). Leia na próxima seção algumas informações que auxiliam na solução do problema.

Informações para solução do problema:

1. Leitura da coleção de cartas: O programa deve ler a coleção de cartas a partir de um documento de texto fornecido, que contém informações sobre os monstros e equipamentos disponíveis.
2. Distribuição inicial de cartas: Cada jogador deve receber aleatoriamente 5 cartas da coleção para formar sua mão inicial.
3. Posicionamento de monstros: A cada rodada os jogadores podem selecionar entre os monstros disponíveis nas cartas na mão e posicionar o monstro no tabuleiro. Os jogadores podem manter posicionados em seu tabuleiro no máximo 5 monstros. Caso o limite de 5 monstros seja

alcançado, o jogador não poderá posicionar mais monstros até que algum seja removido/destruído. Ao posicionar um monstro no tabuleiro, o usuário pode escolher se o mesmo será inserido em estado de ataque ou defesa. O valor atual do monstro depende exclusivamente do estado ao qual está no momento.

   a. Ex: Um Monstro com 2500 de ataque e 1700 de defesa. Se o mesmo está em estado de ataque vamos considerar o valor de 2500. Se o monstro está em estado de defesa vamos considerar o valor de 1700.
   b. Um monstro pode realizar ataques apenas se estiver em estado de ataque.

4. Utilização de equipamentos: Os jogadores podem equipar seus monstros com cartas de equipamento disponíveis na coleção. Os equipamentos podem incrementar tanto o ataque quanto a defesa dos monstros a depender do equipamento utilizado. A cada rodada o jogador pode utilizar apenas uma carta para equipar um de seus monstros posicionados no tabuleiro.
5. Limite de cartas na mão: Cada jogador pode ter no máximo 10 cartas em sua mão. Caso esse limite seja ultrapassado, o jogador deve descartar cartas suficientes para não exceder o limite.
6. Realização de rodadas: O jogo é realizado em rodadas. A cada rodada, os jogadores recebem uma nova carta em suas mãos e escolhem suas ações.
7. Ataques: Durante um ataque, o jogador deve escolher qual monstro do oponente deseja atacar. Se o monstro do oponente estiver na posição de ataque, o oponente perde pontos relativos à diferença de ataque entre os monstros. Se o monstro do oponente estiver na posição de defesa e a defesa for menor que o ataque do jogador, o monstro é destruído e o oponente não perde pontos de ataque. Caso a defesa seja maior que o ataque, o jogador atual(“atacante”) perde pontos.
   a. Quando um monstro realiza um ataque, o usuário ficará impedido de alterar o estado do mesmo na rodada corrente. Podendo alterar o estado apenas na sua próxima rodada.
8. Restrição de ataque: Cada monstro posicionado no tabuleiro só pode atacar uma vez a cada rodada.
9. Cada jogador deve iniciar o jogo com 10000 pontos. Caso a pontuação do jogador seja igual a 0 (zero), o oponente será o vencedor. Caso o baralho não tenham mais cartas, o jogador com a maior pontuação será considerado o vencedor.
10. Caso o usuário não tenha posicionado nenhum monstro no tabuleiro, seu adversário poderá infringir dano direto aos seus pontos.
    a. Os ataques podem ser realizados apenas a partir da segunda rodada.
11. **Você deve utilizar o projeto enviado em anexo**. Qualquer dúvida faça contato.

O que deve ser entregue:

1. Código fonte do programa em Kotlin (bem *indentado* e comentado);
2. Documentação do trabalho. Entre outras coisas, a documentação deve conter:
   2.1. Introdução: descrição do problema a ser resolvido e visão geral sobre o funcionamento do programa;
   2.2. Implementação: descrição sobre a implementação do programa. Deve ser detalhada a estrutura de dados utilizada (de preferência com diagramas ilustrativos), o funcionamento das principais funções e procedimentos utilizados, o formato de entrada e saída de dados, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado;
   2.3. Conclusão: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação;
   2.4. Bibliografia: bibliografia utilizada para o desenvolvimento do trabalho, incluindo sites da Internet se for o caso.
3. Formato: mandatoriamente em PDF.

Obs1: Apesar desse trabalho ser bem simples, a documentação pedida segue o formato da documentação que deverá ser entregue nos próximos trabalhos.

Obs2: Consulte as dicas do Prof. Nívio Ziviani de como deve ser feita uma boa implementação e documentação de um trabalho prático: https://saulocabral.pagekite.me/roteirotp.pdf

Como deve ser feita a entrega:

A entrega **DEVE** ser feita por email na forma de um único arquivo *zipado*, contendo o código, os arquivos, o executável e a documentação.

Comentários Gerais:

- Comece a fazer este trabalho logo, enquanto o problema está fresco na memória e o prazo para terminá-lo está tão longe quanto jamais poderá estar;
- Clareza, *indentação* e comentários no programa também vão valer pontos;
- O trabalho pode ser realizado em duplas (**MÁXIMO de DOIS alun@s**);
- Trabalhos copiados (e **FONTE**) terão nota ZERO;
- Trabalhos entregue em atraso serão aceitos, todavia a nota atribuída ao trabalho será zero;
- Evite discussões inócuas com o professor em tentar postergar a data de entrega do referido trabalho.