MC322 CD – Programação Orientada a Objetos
Especificação do Projeto Final

Leonardo Montecchi (Professor)
Junior Cupe Casquina
Fillipe dos Santos Silva
Natan Rodrigues de Oliveira
Caio Henrique Pardal
Leonardo Yoshida

1s2021

1 Introdução

Legends of Runeterra[1] é um jogo de cartas colecionáveis gratuito desenvolvido e publicado pela Riot Games. Foi lançado em 28 abril de 2020 para dispositivos Android, iOS e Microsoft Windows. Lançado em beta em 24 de janeiro de 2020, situa-se no mesmo universo de League of Legends, o jogo MOBA da Riot Games[2].

As partida são no formato 1x1 (um jogador contra o outro) em uma "mesa" onde as cartas são colocadas em jogo (Figura 1). Cada jogador possui um próprio baralho de no máximo 40 cartas, e começa o jogo com uma mão de 4 cartas retiradas aleatoriamente desse baralho. Cada jogador possui um "Nexus" com 20 pontos de vida inciais. O objetivo é destruir o Nexus do adversário: o jogo termina quando o Nexus de um dos jogadores é levado para zero pontos de vida ou menos.

Figura 1: Imagem do jogo Legends of Runeterra.

[1] https://playruneterra.com/
[2] https://pt.wikipedia.org/wiki/Legends_of_Runeterra

O jogo evolui em rodadas. A cada rodada, um jogador recebe o marcador de ataque e o outro recebe o marcador de defesa. Após cada rodada os marcadores se invertem. Isto é, quem estava atacando fica defendendo e vice-versa. Para jogar uma carta é necessária uma certa quantidade de *mana*; os jogadores recebem gemas de mana no começo de cada rodada.

Existem três tipos de cartas: Campeões, Seguidores, Feitiços. Os campeões são as cartas mais poderosas do jogo, e cada uma possui um critério que, uma vez cumprido, aumentará o nível da carta para uma versão mais poderosa. As cartas de Campeões e Seguidores possuem um valor de *poder* (o dano causado no combate), e um valor de *vida*, sendo a carta destruída se os pontos de vida chegarem a zero. Exemplos dos três tipos de cartas existentes são mostrados na Figura 2.

(a) Campeão (b) Seguidor (c) Feitiço

Figura 2: Exemplos dos diferentes tipos de cartas usadas no jogo.

Uma vez jogadas, as cartas de Seguidores e Campeões "evocam" as respectivas personagens. Para indicar isso, as cartas são colocadas em uma seção especial da mesa do jogador e denotadas por "unidades evocadas". Ao atacar ou defender, o jogador escolhe quais das unidades evocadas participarão na batalha. Uma unidade ataca primeiro a unidade do inimigo que está a sua frente. Caso não tenha nenhuma unidade a sua frente, a unidade ataca o Nexus do opositor. Se o dano recebido por uma unidade for maior ou igual aos pontos de vida da carta, esta será destruída e a carta removida da seção "unidades evocadas".

Mais informações sobre as regras completas do jogo, exemplos de cartas e possíveis efeitos podem ser encontrados no site oficial do jogo, assim como em

sites da comunidade de *fans*[3].

2 Especificação

O objetivo deste projeto é **desenvolver uma adaptação do jogo Legends of Runeterra em Java, aplicando os princípios de programação orientada a objetos**.

Premissas. Para facilitar a implementação, podem ser consideradas as seguintes simplificações:

- A saída do jogo será de forma textual, organizada como de preferência da equipe.
- A entrada será recebida pelo teclado.
- A atualização da tela será feita de forma cíclica. Isto é, a cada t segundos ou depois de ter lido a entrada do usuário. Para tal propósito, pode-se considerar como base o esqueleto de código mostrado na Figura 3.

```java
public class Runner {

    public static void main(String[] args) {

        Game g = new Game();
        g.start();
    }
}
```

```java
public class Game {

    private boolean exitSelected;

    public void start() {
        exitSelected = false;
        System.out.println("Game started!");

        while(!exitSelected) {
            drawBoard();
            readInput();
            updateBoard();
        }
        System.out.println("Game terminated. Bye!");
    }
}
```

Figura 3: Esqueleto de código para o ciclo principal do jogo.

[3] https://lor.mobalytics.gg/pt_br/wiki

Funcionalidades. O jogo deve implementar as seguintes funcionalidades:

1. Os dois jogadores começam a partida com um determinado *deck* (baralho), contendo no máximo 40 cartas. O baralho é embaralhado aleatoriamente a cada partida.

2. No começo da partida, cada jogador recebe 4 cartas selecionadas aleatoriamente do baralho dele. O jogador pode escolher substituir de 0 a 4 cartas, sem saber quais as substituições até a confirmação.

3. O jogo avança em *rodadas*. A cada rodada os jogadores se alternam entre ser *atacante* e *defensor*. Isto é, quem estava atacando fica defendendo e vice-versa. No começo da rodada cada jogador pega uma carta do topo do próprio deck e esta vai para sua mão.

4. Cada rodada evolui em um certo número de *turnos*. O atacante começa o primeiro turno da rodada.

5. No próprio turno, cada jogador poderá "comprar" uma carta entre as que estão na sua mão. Ou seja, descartar uma carta da sua mão, usando pontos de *mana*. Os efeitos da carta, se houver, serão ativados e para cartas Campeões e Seguidores a carta será adicionada na seção de unidades evocadas. Cada jogador pode comprar no máximo uma carta por turno.

6. Após ser compradas, cartas de Campeão e de Seguidores ficam disponíveis como *unidades evocadas* para o jogador. A qualquer momento podem existir apenas 6 unidades evocadas por jogador. É possível substituir uma carta em jogo por uma carta da mão, pagando *mana* apenas pela diferença.

7. No seu turno, o atacante pode decidir *atacar*. Seleciona então entre as unidades evocadas as que participarão na batalha (pelo menos uma).

8. Se o atacante atacar, o defensor pode decidir se *defender*. Seleciona então entre as unidades evocadas as que participarão na batalha. O defensor escolhe também com qual unidade do inimigo cada uma lutará. O defensor pode também decidir não se defender.

9. A *rodada* termina depois da execução do combate, ou se o atacante decidir terminar a rodada sem atacar.

10. *Mana*. Os jogadores começam o jogo com 0 pontos de mana. No começo de cada rodada cada jogador recebe n pontos de mana, sendo $n = 1$ na primeira rodada. O valor de n incrementa a cada rodada, até um máximo de $n = 10$. Ou seja, recebem 1 mana na primeira rodada, 2 mana na segunda, etc., 10 mana na decima rodada e 10 nas seguintes. Até um máximo de 3 pontos de mana que não foram usados podem ser guardados para a rodada seguinte, como "mana de feitiço", que pode ser usado apenas para comprar cartas de feitiço. O resto do mana que não foi usado na rodada é perdido.

11. *Combate*. Se houver combate, esse encerra a rodada e o jogo passa então à rodada seguinte. O combate acontece entre as unidades selecionadas para o combate pelos dois jogadores. Exceto efeitos especiais das cartas, o combate é executado da seguinte forma:

    - Se uma unidade do defensor foi escolhida para se opor a uma unidade do atacante, elas combatem entre si. Pontos de *poder* de uma são subtraídos aos pontos de *vida* da outra, e vice-versa. Se uma unidade ficar com 0 ou menos pontos de vida é removida do jogo.
    - Se nenhuma unidade do defensor se opõe a uma unidade do atacante, esta então ataca diretamente o Nexus. Os pontos de *poder* da unidade são subtraídos aos pontos de vida do Nexus do defensor.
    - Unidades do defensor não podem atacar o Nexus do atacante.
    - Unidades que sobrevivem ao combate voltam entre as unidades evocadas do jogador. Eventuais diminuições aos pontos de vidas permanecem nas rodadas seguintes, exceto efeitos especiais das cartas ou cartas de feitiço.

12. *Cartas*. Como mencionado acima, existem três tipos de cartas: Campeões, Seguidores e Feitiços. Todas as cartas possuem um *nome* e um *custo* em mana. Cartas que evocam unidades (Seguidores ou Campeões) possuem o valor de *poder* e de *vida* da unidade. Algumas cartas possuem *efeitos* especiais, ou seja, ações que são aplicadas ao estado atual do jogo.

    - *Seguidores*. Cartas de seguidores possuem um valor de *poder* e de pontos de *vida*. As cartas seguidores podem ter efeitos que se aplicam na hora da evocação (compra da carta), ou no final do turno do

adversário. Seguidores podem também possuir *traços*, que modificam parcialmente o comportamento do seguidor.

- *Campeões*. Cartas de campeões podem ser vistas como cartas de seguidores especiais. Além das características dessas, elas possuem um nível que muda durante o jogo, e pode ser "normal" ou "superior". Cada campeão tem uma condição para subir de nível, que pode ser:
  - Atacar n vezes
  - Matar n seguidores do inimigo
  - Fazer n pontos de dano
  - Sofrer n pontos de dano

  Subir ao nível "superior" faz com que o campeão receba uma ou mais melhorias, entre:
  - Aumentar de n o seu *poder*
  - Aumentar de n a sua *vida*
  - Ganhar um traço ou efeito a mais
- *Feitiços*. Os efeitos das cartas de feitiço tem ação imediata na hora da compra da carta. Depois do uso, as cartas de feitiço são descartadas (não ficam entre as unidades evocadas). As cartas de feitiço podem ser compradas usando os três pontos de mana extra de feitiço, eventualmente complementando com pontos de mana normais.

13. *Traços*. Os traços possíveis para as cartas Seguidores e Campeões são os seguintes.
    - *Ataque duplo*. Ataca duas vezes no mesmo turno.
    - *Elusivo*. Pode ser bloqueado apenas para outra carta elusiva.
    - *Fúria*. Quando destruir um seguidor adversário ganha $+n/+m$.
14. *Efeitos*. Os efeitos possíveis para as cartas são os seguintes. Vale destacar que uma carta pode, no geral, ter mais que um efeito.
    - Dê $+n/+m$ a todas as unidades aliadas evocadas.
    - Dê $+n/+m$ a uma unidade aliada nesta rodada.
    - Se a carta destruir uma unidade do inimigo nesta rodada, é colocada uma nova carta de uma unidade especifico na sua mão.

- Cure inteiramente uma unidade aliada.
- Dobre o ataque e defesa de uma unidade aliada.
- Escolha um aliado e um oponente para um combate imediato.
- Uma unidade evocada ataca o Nexus do adversário
- Um aliado atacante golpeia todos os oponentes defensores.
- Ao ser destruído, você ganha uma carta.
- Altera o poder de uma unidade para 0 nesta rodada.
- Cria uma barreira que anula o próximo dano que uma unidade aliada levaria. Dura uma rodada.
- Golpeia o Nexus do adversario para n pontos de dano.

Uma lista mais extensa dos efeitos existentes no jogo real pode ser encontrada neste link: `https://lor.mobalytics.gg/pt_br/wiki`.

15. O jogo deve suportar jogadores humanos e artificiais (dois humanos, dois computadores, ou humano vs computador). Considere as seguintes simplificações:
    - O jogador artificial (computador) joga de forma aleatória, escolhendo aleatoriamente entre as cartas disponíveis.
    - No caso de dois jogadores humanos, eles compartilham o mesmo terminal (não importa se um consegue ver as cartas na mão do outro).
16. A cada turno, o sistema deve mostrar a situação atual da mesa e listar as ações possíveis para o(s) jogador(es) humano(s). O sistema deve também destacar as cartas que podem ser jogadas no turno atual entre as que estão na mão do jogador. Caso o jogador queira, o sistema deve mostrar o detalhamento das cartas na sua mão; isto é, a lista de efeitos e traços. Efeitos de compras de cartas e evolução do combate devem também ser mostradas para o usuário.
17. *Gerenciamento de Decks*. O sistema deve ser extensível na forma de criação e uso de *decks*. Isto é, deve ser possível implementar novas formas de criar baralhos sem alterar a estrutura principal do sistema. *Sugestão*: Uma forma de fazer isso é usar o padrão de projeto *Factory* ou *Factory Method*[4].

[4]Na hora da disponibilização deste texto não teremos visto padrões de projeto ainda. Padrões de projetos são formas estabelecidas (padrões) para organizar o código ao fim de resolver um determinado problema conhecido.

18. *Deck Base.* O *deck* base "Demacia" é formado pelas seguintes cartas. A notação c/a/d está por custo (em mana) / ataque (poder) / defesa (pontos de vida).

    - *Campeões*
        - *Nome*: Garen. *Stats*: 5/5/5. *Efeito*: Se cura totalmente no final de cada rodada. *Level Up*: Depois de ter atacado duas vezes. *Evolução*: Ganha o traço "Elusivo"; +1 ao poder; +1 pontos de vida.
    - *Seguidores*
        - *Nome*: Tiana. *Stats*: 8/7/7. *Efeito*: ao ser comprada: uma unidade evocada ataca o Nexus do adversário.
        - *Nome*: Vanguarda. *Stats*: 4/3/3. *Efeito (Evocação)*: Dê +1/+1 a todos os seguidores aliados.
        - *Nome*: Duelista. *Stats*: 3/3/2. *Efeito*: Se a carta destruir um seguidor do inimigo nesta rodada, uma carta "Poro" é colocada em sua mão.
        - *Nome*: Defensor. *Stats*: 2/2/2. *Traço*: Fúria (+0/+1).
        - *Nome*: Poro. *Stats*: 1/2/1
        - *Nome*: Poro Defensor. *Stats*: 1/1/2. *Efeito*: Ao ser destruído, você ganha uma carta.
    - *Feitiços*.
        - *Nome*: Julgamento. *Custo*: 8. *Efeito*: Um aliado atacante golpeia todos os oponentes defensores.
        - *Nome*: Valor Redobrado. *Custo*: 6. *Efeito*: Cure inteiramente um aliado; Dobre o ataque e defesa deste aliado.
        - *Nome*: Golpe Certeiro. *Custo*: 1 *Efeito*: Dê +1/+1 a um aliado nesta rodada.
        - *Nome*: Combate um-a-um. *Custo*: 2. *Efeito*: Escolha um aliado e um oponente para um combate imediato.

Comportamentos que não foram especificados nesta lista podem ser decididos pelas equipes ou colocados como parâmetros de configuração do jogo. Por exemplo, o que acontece quando um dos jogadores acabar as cartas do seu *deck*. Escolhas deste tipo devem ser documentadas no relatório do projeto e/ou com comentários no código.

Funcionalidades Adicionais. É permitido a inclusão de funcionalidades adicionais (ex: interface gráfica, limite de tempo para as jogadas, organização cliente/servidor para dois jogadores), desde que as funcionalidades básicas listadas acima sejam realizadas corretamente.

Relatório. Junto com o código fonte deve ser produzido um breve relatório que explique as funcionalidades implementadas e as escolhas tomadas durante o desenvolvimento. O propósito deste relatório é explicar porque foi decidido organizar o código de uma certa forma, quais problemas foram encontrados e como eles foram resolvidos e, eventualmente, detalhar funcionalidades adicionais que foram incluídas. O relatório deve conter um diagrama UML das principais classes implementadas e as relações entre elas.

O nível de detalhamento do relatório é deixado ao bom senso da equipe. Observe que relatório serve para reduzir o tempo necessário para entender o que foi feito. Um texto de 50 páginas ou um diagrama com todos os detalhes de todas as classes (mesma informação que já está no código) não ajuda muito neste sentido.

3 Equipe

O projeto deve ser desenvolvido **em equipe de 2-3 alunos**. Os integrantes de cada equipe devem ser sinalizados na plataforma Google Classroom, por meio da atividade "Definição das Equipes do Projeto". Eventuais alterações das equipes devem ser discutidas com o professor. Fiquem à vontade para usar os canais da disciplina (Classroom, Discord) para achar integrantes caso seja necessário.

É recomendado o uso da ferramenta de controle de versão **git**[5] (ou equivalente) para facilitar a integração do código desenvolvido remotamente por cada integrante. Existem plataformas que permitem criar e manter repositórios git online, como o GitHub ou o GitLab. A Unicamp também mantém uma instalação institucional do GitLab[6].

4 Submissão

O projeto deve ser submetido no link de entrega no Google Classroom da disciplina, junto com um breve relatório que detalhe as escolhas de implementação

[5] `https://guides.github.com/introduction/git-handbook/`
[6] `https://gitlab.unicamp.br`

e qualquer informação necessária para executar a aplicação. **A submissão deve ser feita em formato de arquivo compactado (zip/tar.gz/rar) e o nome do arquivo deve estar no formato {NomeGrupo}_Projeto**, onde {NomeGrupo} deve ser substituído pelo nome da própria equipe.

- O arquivo deve conter:
  1. O código fonte do projeto.
  2. Um breve relatório explicando o funcionamento do programa e as escolhas efetuadas no projeto do código (classes, relações, etc.), em formato pdf.
- Quando: Até dia **10 de julho de 2021**.

5 Avaliação

A nota base do projeto será entre 0 e 10. Faz parte da avaliação decidir quais classes devem ser criadas e em quais classes cada método deve ser implementado. Em particular, o código será avaliado de acordo com os seguintes aspectos:

1. Definição de classes (30%).
2. Aplicação de princípios de POO (30%).
   - Ex: *encapsulamento* de atributos, *responsabilidades* de classes, aplicação de *herança*, uso de *polimorfismo*.
3. Funcionalidades implementadas (30%).
4. Relatório (10%).

Apenas no caso em que todas as funcionalidades requeridas forem implementadas corretamente será considerado o acréscimo de até um (1) ponto para funcionalidades adicionais implementadas.

Isto é, a nota final do projeto pode atingir até 11 pontos. Funcionalidade "implementada corretamente" significa 1) funcionando corretamente e 2) implementada de acordo com os princípios de programação orientada a objetos.

6 Fraudes

Qualquer tipo de fraude acarretará em nota final zero para todos os envolvidos. Exemplos de fraudes são:

- compartilhar código não trivial entre equipes diferentes;
- copiar código não trivial da Internet;
- comprar implementações prontas.