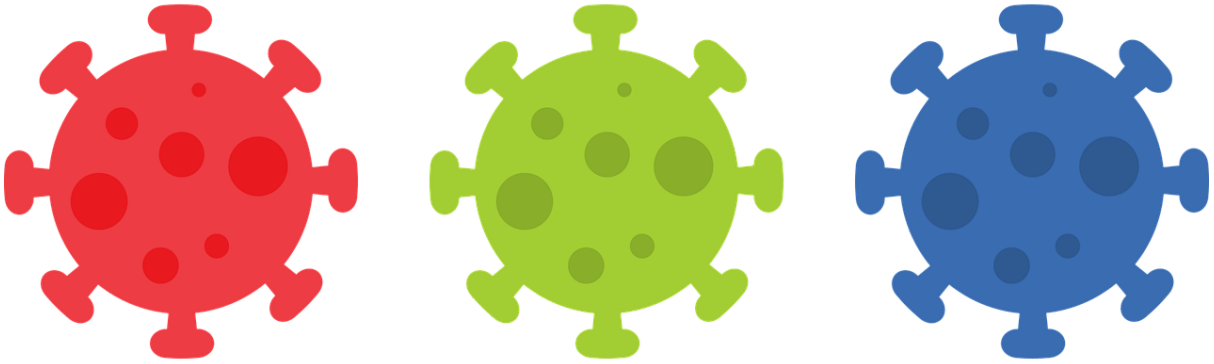
Universidade Federal do Paraná
Centro Politécnico
CI1062 - Paradigmas de Programação
CI062 - Técnicas Alternativas de Programação
Profª Rachel Reis

Trabalho Prático (30 pontos)

"Antivírus por um dia"



1. Introdução

Antivírus por um dia é um jogo de tabuleiro *desktop* colaborativo, projetado para dois jogadores. Esse jogo tem como objetivo percorrer os setores da memória de um computador infectado por vírus. O jogador assume o papel do antivírus, que deve avançar pelos setores, eliminando os vírus que encontrar, até que o setor com a fonte de infecção seja alcançado. Mas tome cuidado, os vírus podem te destruir no caminho.

2. Regras do Jogo

O jogo possui dois personagens principais, os **jogadores** e os **inimigos** (vírus), e é baseado em um sistema de **turnos** para combate. Mais informações sobre o jogo são apresentadas nas próximas subseções.

2.1. Jogadores

O jogo é formado por dois tipos de jogadores: simples e de suporte.

1. Cada jogador possui um valor para ataque (ATK) e um valor de defesa (DEF), sendo que o poder de ataque e defesa dos jogadores são fixos. O jogador simples deve ter ATK=2 e DEF=6 (representado por 2/6) e o jogador de suporte deve ter ATK=1 e DEF=7 (representado por 1/7).
2. Cada jogador ocupa uma posição no tabuleiro (POS), com duas coordenadas (X,Y).
3. O jogador simples não possui habilidades especiais, já o jogador de suporte, possui a capacidade de recuperar 2 pontos de sua defesa ou da defesa de outro jogador que esteja no mesmo setor.
4. O jogador de suporte deve sempre possuir ataque menor que o jogador simples.
5. O jogador de suporte não pode existir sem um jogador simples.
6. O jogador simples pode executar as seguintes ações no jogo: movimentar, atacar e procurar.
7. O jogador de suporte pode executar as seguintes ações no jogo: movimentar, atacar, procurar e recuperar defesa.
8. As ações dos itens 6 e 7 serão detalhadas na seção **Turno**.

2.2. Inimigo

Os inimigos são os vírus de computador gerados por uma fonte de infecção. O objetivo do jogo é que um dos jogadores, apresentados na Seção 2.1, encontre essa fonte de infecção.

1. Cada inimigo possui um valor para ataque (ATK) e um valor de defesa (DEF), sendo que o poder de ataque e defesa devem ser inicialmente iguais, com valores entre 1 e 3. Por exemplo, um inimigo pode ter ATK=1 e DEF=1 (representado por 1/1), e outro pode ter ATK=2 e DEF=2 (representado por 2/2).
2. Cada inimigo ocupa uma posição no tabuleiro (POS), com duas coordenadas (X,Y).
3. Os valores de ATK e DEF devem ser definidos aleatoriamente, no momento em que os inimigos forem criados.
4. Os inimigos só podem executar a ação de atacar, que será detalhada na seção **Turno**.
5. Os inimigos não se movimentam, não efetuam procuras ou recuperam defesa (somente os jogadores possuem essas ações).

2.3. Tabuleiro do jogo

O cenário do jogo é formado por um tabuleiro 5x5, conforme ilustrado na Figura 1.

Abaixo seguem as informações sobre o tabuleiro mostrado na Figura 1:

1. Os jogadores iniciam o jogo no centro do tabuleiro (posição C).
2. A fonte de infecção de vírus (posição X), que consiste no objetivo final, deve ser aleatoriamente posicionada em qualquer posição do tabuleiro que não seja a central.
3. Cada posição do tabuleiro corresponde a um setor que o jogador deve percorrer, até encontrar a fonte de infecção do vírus (X).
4. Cada setor possui uma posição no tabuleiro [x,y] e quatro lados que podem ser portas ou paredes. Por exemplo, na Figura 1, a posição C corresponde ao Setor [3,3], que possui quatro portas (*) e dois jogadores: P1 (jogador simples) e P2 (jogador de suporte).

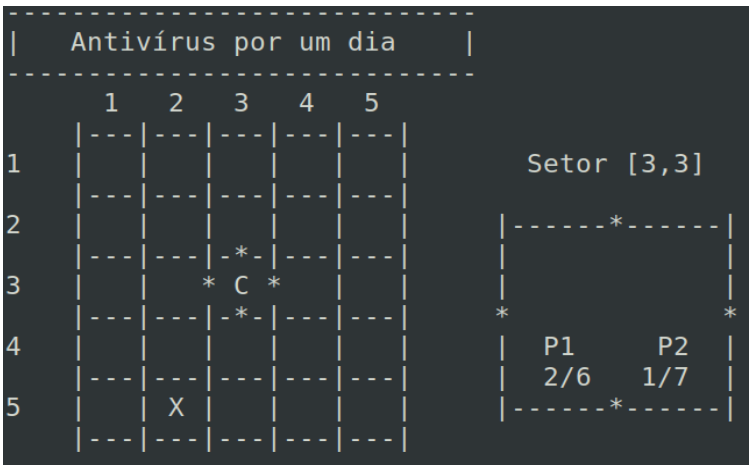


Figura 1 – Tabuleiro inicial do jogo “Antivirus por um dia”.

5. Cada setor possui uma posição no tabuleiro [x,y] e quatro lados que podem ser portas ou paredes. Por exemplo, na Figura 1, a posição C corresponde ao Setor [3,3], que possui quatro portas (*) e dois jogadores: P1 (jogador simples) e P2 (jogador de suporte).
6. Cada setor pode possuir até 4 portas para que um jogador se movimente entre os setores. Estas portas serão definidas pelo programador durante o desenvolvimento, ou seja, em tempo de compilação.
7. Os limites do tabuleiro (parte externa) são sempre paredes, ou seja, não podem possuir portas.
8. A posição inicial (C) sempre possui as 4 portas abertas, que estão representadas na Figura 1 por um asterisco (*).
9. Podem existir 3 tipos de setores:
    a. **Setor normal**: Não existe nenhuma restrição, ou seja, todas as ações dos jogadores P1 e P2 podem ser executadas. Os inimigos sempre recebem o dano de um ataque e a procura pode ser efetuada normalmente.
    b. **Setor oculto**: A restrição neste setor é que quando o jogador efetuar um ataque, o vírus pode ou não ser eliminado (aleatório), pois o jogador não sabe a localização exata do vírus no setor.
    c. **Setor privado**: A restrição é que os jogadores não podem executar a ação de procurar no setor.

2.4. Turno

O jogo deve funcionar em sistema de turnos, alternando entre jogadores e inimigos.

1. Os jogadores possuem até 25 **ciclos** para encontrar a fonte de infecção do vírus (X). Cada ciclo é formado pelos turnos do P1, P2 e inimigos.
2. Os jogadores iniciam no centro do tabuleiro (Turno 0) e devem selecionar uma das portas do setor inicial (C) para se movimentar para um novo setor. O primeiro a jogar é sempre o jogador simples (P1), seguido pelo jogador de suporte (P2).
3. Quando um jogador se movimentar para um setor ainda não visitado, duas ações devem ocorrer:
    a. As portas do tabuleiro do setor visitado devem se tornar visíveis.

b. Podem ser criados de 1 a 3 inimigos no setor, com ATK igual a DEF e no intervalo de 1 a 3. Tanto o número de inimigos, quanto o seu ATK e DEF devem ser gerados aleatoriamente.

Por exemplo, na Figura 2, os jogadores P1 e P2 se movimentaram para o mesmo setor ([3,4]), que foi criado com duas portas e uma parede, mantendo a porta do setor anterior ([3,3]). Também foram gerados 3 inimigos com ATK/DEF iguais a 1/1 (inimigo 1), 3/3 (inimigo 2) e 2/2 (inimigo 3).

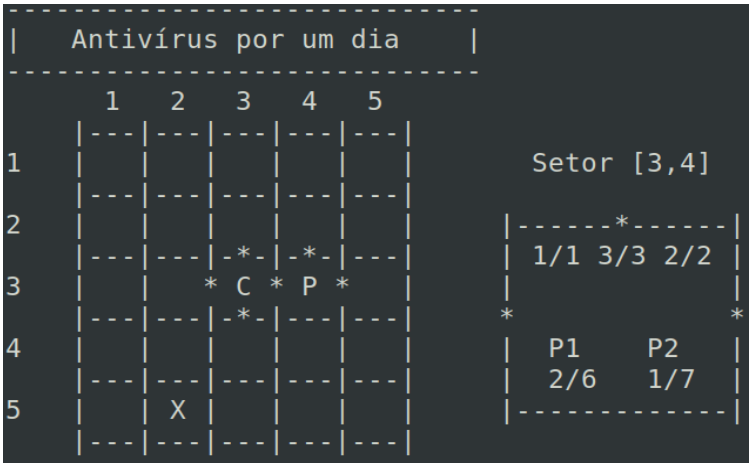


Figura 2 – Jogadores P1 e P2 se movem para o mesmo setor.

4. Após entrar no setor e gerar os inimigos, o turno inicia com o jogador P1, que pode executar duas ações*:
    a. **Atacar** um inimigo. Por exemplo, suponha que P1 com ATK=2 ataque o inimigo 2 (3/3). Esse ataque reduz em 2 a DEF do inimigo, ou seja, o inimigo 2 ficará com ATK/DEF igual a 3/1.
    b. **Procurar** no setor por itens. Esta procura consiste em sortear um número aleatório entre 1 e 6.
        i. Para números entre 1 e 3, nada é encontrado no setor.
        ii. Para número igual a 4, o jogador ganha 1 de DEF.
        iii. Para número igual a 5, o jogador ganha 2 de DEF.
        iv. Para número igual a 6, todos os inimigos do setor perdem 1 de DEF.

    *Cada opção acima (a,b) representa uma única ação, ou seja, o jogador pode: atacar duas vezes, atacar e procurar ou procurar duas vezes.

5. Após as duas ações do jogador P1, o turno do jogador P2 é iniciado. No seu turno, P2 pode realizar as mesmas ações descritas no item 4. Além disso, o jogador P2 pode optar por usar sua habilidade especial (recuperar defesa) no lugar das ações de atacar e procurar. Essa habilidade consiste em adicionar 2 a DEF dele ou do jogador P1, desde que P1 esteja no mesmo setor que P2.
6. Após finalizar os turnos de P1 e P2, os inimigos que estiverem vivos iniciam seu turno de ataque (única ação permitida). Cada inimigo ataca cada jogador uma única vez (somente os jogadores que estiverem no mesmo setor). Por exemplo, suponha que o inimigo 3 (2/2) ataque P1 (2/6) e P2 (1/7). Para cada ataque, um número

aleatório entre 1 e 6 deve ser gerado. Se o número for par, o ataque é realizado; caso contrário, nada acontece. No caso do ataque ser destinado a P1, ele ficará com 2/4, ou seja, a DEF do jogador atacado é reduzida pelo ATK do inimigo atacante. Em seguida, o inimigo 3 deve atacar P2 usando esse mesmo raciocínio.

7. Após todos os inimigos do setor finalizarem seu ataque, um novo ciclo é iniciado.
8. Os jogadores P1 e P2 só podem se movimentar para outro setor, quando não existirem mais inimigos no setor em que eles se encontram. A Figura 3 mostra um exemplo em que P1 se movimentou para o Setor [2,4] e P2 para o Setor [3,5], onde as portas ficaram visíveis e os inimigos foram gerados.

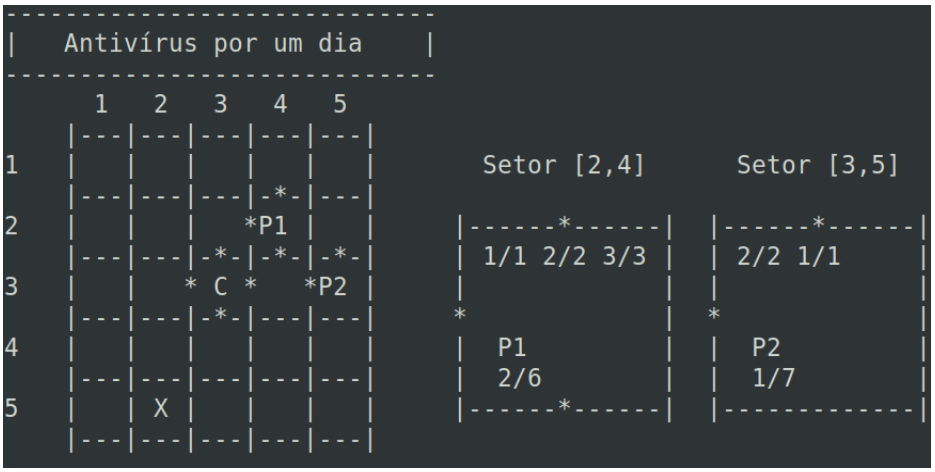


Figura 3 – Jogadores P1 e P2 se movem para setores diferentes.

9. Setores já visitados não geram novos inimigos, porém ao passar por este setor, o jogador pode usar o seu turno para **procurar** itens (permitido aos jogadores P1 e P2) ou **recuperar defesa** (permitido apenas ao jogador P2).
10. O jogo pode terminar de três formas:
    a. Um jogador (P1 ou P2) entra no setor que se encontra a fonte de infecção (X). Nesse caso, o jogador que entrar no setor será o vencedor do jogo.
    b. Quando exceder o número de 25 ciclos.
    c. Quando todos os jogadores forem eliminados pelos inimigos (DEF <= 0).

3. Avaliação

A avaliação consiste na entrega do código do projeto e relatório em formato PDF. Os alunos que não enviarem o código fonte ou relatório receberão automaticamente a nota **zero**.

Para avaliação do trabalho, o sistema deverá atender aos conceitos da orientação à objetos listados abaixo:

1. Classes (atributos e métodos)
2. Construtores
3. Encapsulamento
4. Herança
5. Interface
6. Classe Abstrata
7. Polimorfismo
8. Coleção

4. Observações importantes

- Grupos de no **MÁXIMO** 04 estudantes.

- Sobre o Trabalho:
  - Deve ser implementado na linguagem Java.
  - Deve conter o nome e GRR dos integrantes do grupo como comentário na classe que contém o método main().
  - Deve ser entregue pelo site **replit.com**, seguindo os seguintes passos:
    1. Criar um usuário no replit.com.
    2. Criar um projeto **Java público** com o nome "TrabalhoCI1062" (sem espaços).
    3. Fazer o upload dos arquivos ".java".
    4. Clicar no nome do trabalho (parte superior esquerda da tela) e no botão Publish.
    5. Na nova janela, adicionar na descrição (Repl description) o primeiro nome e GRR de todos os membros do grupo e clicar nos botões "Next" até surgir o botão "Publish to Community".
    6. Copiar o link da "Cover Page" e adicionar ao relatório.

- Sobre o Relatório:
  - Deve ser entregue em um arquivo .pdf nomeado com o GRR de um dos membros do grupo. Ex.: grr1.pdf.
  - Deve conter o nome completo de todos os membros do grupo seguido do respectivo GRR.
  - Deve ser entregue via Moodle C3SL (https://moodle.c3sl.ufpr.br/login/index.php) na área da disciplina de CI1062 - Paradigmas de Programação.
  - Deve ter no máximo duas páginas, utilizando espaçamento simples e coluna dupla, ou no máximo três páginas com espaçamento 1,5 ou duplo e formato de uma coluna.
  - Deve mostrar e/ou explicar onde foram usados os conceitos da orientação a objetos, principalmente os conceitos de Herança, Interface, classe Abstrata, Polimorfismo e Coleção. Na apresentação do conceito polimorfismo é importante especificar o tipo que foi utilizado.
  - Pode incluir diagramas ou outro recurso visual que julgar necessário, de forma a tornar o relatório mais informativo. Não adicione imagens com o print do código. Caso queira dar destaque a alguma parte da implementação, copie o trecho de código e adicione a explicação.

- O trabalho deverá estar disponível no Replit e o relatório entregue no Moodle C3SL até **15/02/2023** às **23:59**. Em nenhuma hipótese serão recebidos trabalhos por outro meio e fora do prazo.

5. Distribuição da Nota

A nota do trabalho é composta por 70% referente a completude e qualidade de implementação, e 30% referente ao relatório. Esses valores podem variar para casos específicos. Por exemplo, os grupos que deixarem de entregar o relatório e/ou a implementação do trabalho receberão nota zero.

Alguns descontos padrão, considerando uma nota entre 0 e 100 pontos para o trabalho e relatório:

- Plágio: perda total da pontuação para todos os envolvidos. Isso é válido mesmo para casos onde o plágio se refere a apenas um trecho do código.
- Arquivos implementados em linguagens diferentes de Java serão desconsiderados.
- Arquivos com problemas ou erros de compilação/execução são de inteira responsabilidade dos grupos: desconto de 5 a 70 pontos.
- Erros e avisos de compilação: desconto de 5 a 70 pontos.
- Entrega do trabalho sem o nome e GRR dos integrantes: desconto de 5 pontos.
- Entrega do relatório no formato e nome incorreto: desconto de 5 pontos.
- Erros e informações inconsistentes no relatório: desconto de 5 a 30 pontos.

6. Dicas

6.1 Gerar números aleatórios

Para gerar números aleatórios no Java utilize a classe Random. Para isso, você deve adicionar a biblioteca: `import java.util.Random`. Abaixo segue um exemplo de código em que cinco números aleatórios são gerados no intervalo de 1 a 10.

```
Random aleatorio = new Random();
int num;
for(int i = 0; i < 5; i++){
    num = aleatorio.nextInt(10);
    System.out.println(num);
}
```

6.2 Adicionar cores ao Terminal

Para adicionar cores às letras e fundo do texto, utilize a classe **Cores** abaixo. Na sequência, segue um exemplo de execução no método main().

```
public class Cores {
  public static final String ANSI_RESET = "\u001B[0m";
  public static final String ANSI_BLACK = "\u001B[30m";
  public static final String ANSI_RED = "\u001B[31m";
  public static final String ANSI_GREEN = "\u001B[32m";
```

```
  public static final String ANSI_YELLOW = "\u001B[33m";
  public static final String ANSI_BLUE = "\u001B[34m";
  public static final String ANSI_PURPLE = "\u001B[35m";
  public static final String ANSI_CYAN = "\u001B[36m";
  public static final String ANSI_WHITE = "\u001B[37m";

  public static final String ANSI_BLACK_BACKGROUND = "\u001B[40m";
  public static final String ANSI_RED_BACKGROUND = "\u001B[41m";
  public static final String ANSI_GREEN_BACKGROUND = "\u001B[42m";
  public static final String ANSI_YELLOW_BACKGROUND = "\u001B[43m";
  public static final String ANSI_BLUE_BACKGROUND = "\u001B[44m";
  public static final String ANSI_PURPLE_BACKGROUND = "\u001B[45m";
  public static final String ANSI_CYAN_BACKGROUND = "\u001B[46m";
  public static final String ANSI_WHITE_BACKGROUND = "\u001B[47m";
}

public class Principal{
  public static void main(String[] args) {
    System.out.println(Cores.ANSI_RED + "Red text" + Cores.ANSI_RESET);
    System.out.println(Cores.ANSI_BLUE + "Blue text" + Cores.ANSI_RESET);
    System.out.println(Cores.ANSI_YELLOW_BACKGROUND + "Yellow background"
                    + Cores.ANSI_RESET);
  }
}
```