MAC 110 – Introdução à Computação
IME-USP – Primeiro Semestre de 2003

Terceiro Exercício Programa (EP3)
Campo Minado

Divulgação do enunciado: 3/6/2003
Prazo de entrega: 27/6/2003

1 Introdução

Um jogo de estratégia muito popular é o Campo Minado (*Minesweeper*). Ele possui diversas implementações, inclusive uma muito famosa que é distribuída com o Windows. Neste EP, o objetivo é criar a nossa própria implementação, construindo um "tabuleiro" para o jogo, assim como "jogadores" automáticos ou não.

Antes de começar o EP, jogue umas partidas na sua implementação favorita para entender bem as regras descritas a seguir.

2 Campo Minado

O jogo consiste de um tabuleiro quadriculado, representado por uma matriz ($n \times m$), e de k minas que estão escondidas em posições aleatórias do tabuleiro. Apenas um jogador participa do jogo.

Inicialmente, o jogador não tem nenhuma informação sobre a localização dessas minas, exceto que cada posição do tabuleiro pode conter no máximo uma mina.

Cabe ao jogador descobrir a localização das minas no tabuleiro. Para isso, em cada jogada, ele escolhe uma posição válida do tabuleiro para ser revelada. Com relação a essa escolha, existem duas possibilidades.

1. A posição contém uma mina. Neste caso, o jogador perde o jogo e o tabuleiro mostra a posição de todas as minas.

2. A posição não contém uma mina. Neste caso, o tabuleiro deve revelar a posição escolhida pelo jogador. A revelação é feita pelo seguinte procedimento.

   - A posição revelada é rotulada com o número de minhas presentes nas oito posições adjacentes a ela. Isto é, o jogador passa a enxergar esse número na posição revelada.
   - Se esse número for zero, toda posição adjacente à posição revelada é também revelada usando-se este mesmo procedimento. Note que, com isso, se uma posição adjacente também não tiver minas nas suas vizinhas, as posições adjacentes a ela também serão reveladas, e assim sucessivamente.

Observação: o jogo se encarrega de garantir que a primeira escolha do jogador nunca será uma posição com mina.

O objetivo do jogo é encontrar as k minas revelando as $n.m-k$ posições da matriz que não contêm minas.

3 O Exercício

O exercício consiste na implementação de (1) uma classe `Tabuleiro`, onde o jogo possa ser jogado, (2) uma classe `JogadorHumano` por onde possa se jogar entrando os dados pelo teclado e (3) uma classe `JogadorAutomatico` que joga automaticamente. Todas as classes devem respeitar as **interfaces**[1] definidas de forma a poderem ser testadas automaticamente.

3.1 O Tabuleiro

A classe `Tabuleiro` deve possuir a seguinte interface:

```
// Construtores:
public Tabuleiro(int n, int m, int k);
public Tabuleiro(int i);

// Métodos de acesso aos atributos:
public int numLinhas();
public int numColunas();
public int numMinas();

// Métodos de usuário:
public int numPosRestantes();
public int numPosReveladas();
public int numJogadas();
public boolean fim();
public boolean ganhou();
public boolean perdeu();
public void recomeca();
public void imprimeTabuleiro();
public int jogada(int i, int j);
```

O construtor `Tabuleiro(int i)` deve criar tabuleiros com n, m e k pré-definidos, simulando os níveis:

- fácil ($n = 8$, $m = 8$ e $k = 10$), se `i` $= 1$;
- médio ($n = 16$, $m = 16$ e $k = 40$), se `i` $= 2$; e
- difícil ($n = 16$, $m = 30$ e $k = 99$), se `i` $= 3$.

O construtor que recebe três inteiros deve criar um tabuleiro com `n` linhas e `m` colunas e com `k` minas. Caso esses valores não sejam viáveis, um tabuleiro de nível fácil deve ser criado.

O método `jogada` deve efetuar a jogada na posição (`i`,`j`) do tabuleiro, conforme descrita na Seção 2. O valor devolvido deve ser -1 se a posição escolhida tiver uma mina (jogador perdeu); 0 se

[1] Respeitar uma interface significa implementar todos métodos pedidos, com assinaturas idênticas às dadas.

todas as posições foram reveladas (jogador venceu); 1 se a posição foi revelada; ou 2 se a posição já tinha sido revelada antes. Se os parâmetros forem inválidos, devolve -2. Depois que o jogador venceu ou perdeu, o método não realiza mais jogadas e devolve sempre o mesmo valor (0 ou -1, dependendo do caso).

Os métodos `numPosReveladas` e `numPosRestantes` devem devolver, respectivamente, o número de posições reveladas até o momento e o número de posições livres que ainda precisam ser reveladas (esses números estão relacionados). O método `numJogadas` devolve o número de jogadas válidas realizadas até o momento.

O método `imprimeTabuleiro` deve imprimir o estado atual do tabuleiro para o jogador, isto é, deve mostrar as posições reveladas até o momento. Se a partida terminou, deve imprimir também as minas.

O método `fim` informa se a partida terminou (devolve `true` se terminou). O método `ganhou` devolve `true` se a partida terminou e o jogador venceu, já o método `perdeu` devolve `true` se a partida terminou e o jogador perdeu.

O método `recomeca` gera um novo tabuleiro e reinicia o jogo como se nenhuma jogada tivesse sido feita.

Os métodos de acesso aos atributos devolvem os valores dos atributos n, m e k. Não é permitido alterar os valores desses atributos, então não implemente métodos para isso.

Dica 1 *Note que os métodos de acesso aos atributos protegem os atributos de alterações feitas de fora da classe. Mas para isso funcionar, você deve declarar os atributos com o modificador* `private`*. Por exemplo:*

```
public class Tabuleiro
{
  private int N;        // número de linhas
  private int M;        // número de colunas
  private int K;        // número de minas

  private int[][] Tab;  // tabuleiro

  ⋮
}
```

Assim não é possível acessar diretamente os atributos:

```
> Tabuleiro t = new Tabuleiro(1);
> t.N
java.lang.IllegalAccessException: Class koala.dynamicjava.interprete
r.EvaluationVisitor can not access a member of class Tabuleiro with
modifiers "private"
  at sun.reflect.Reflection.ensureMemberAccess(Reflection.java:57)
  at java.lang.reflect.Field.doSecurityCheck(Field.java:811)
  at java.lang.reflect.Field.getFieldAccessor(Field.java:758)
  at java.lang.reflect.Field.get(Field.java:228)
```

Isso é especialmente desejável neste EP, já que não queremos, por exemplo, que o jogador trapaceie olhando a representação interna do tabuleiro.

Mas não só neste EP, em geral também é interessante proteger todos os atributos das classes. Por isso, ***sempre*** *declare os atributos de qualquer classe como* `private`*, e forneça os métodos de leitura*

ou escrita conforme necessário. ***Nunca*** *acesse os atributos diretamente se o código estiver fora da classe.*

Dica 2 *Como sempre, sinta-se à vontade para criar métodos adicionais na sua classe. Mas lembre-se de que as outras classes só podem conhecer e utilizar os métodos definidos na interface acima. Ou então os métodos adicionais podem ser usados para a depuração e testes do seu programa. Isso significa que eles podem ser usados apenas para descobrir erros e nunca no funcionamento normal do seu programa. Por exemplo, você poderia criar o método* `void depImprimeTabuleiro()` *que imprime a representação interna do seu tabuleiro, exibindo inclusive as posições das minas. Isso seria muito útil para descobrir erros. Outro método poderia ser o* `Coordenada[] depDescobreMinas()` *que devolve um vetor com as posições de todas as minas (*`Coordenada` *seria uma classe extra).*

3.1.1 Representação do Tabuleiro

Há diversas maneiras de se representar o tabuleiro com as minas e as jogadas. Nesta seção sugerimos uma maneira.

O tabuleiro é representado por uma matriz de inteiros. Cada posição da matriz armazena um valor que representa o estado da posição correspondente do tabuleiro. Os possíveis valores são:

- -2: a posição contém uma mina;
- -1: a posição não contém uma mina e não foi revelada ainda;
- i, com $0 \leq i \leq 8$: a posição não contém uma mina e já foi revelada. Além disso, existem i minas distribuídas nas 8 posições adjacentes a ela.

Dica 3 *Ao criar a matriz, crie também uma* moldura *(ou* borda*) que delimita as fronteiras do tabuleiro. Isto é, em vez de criar uma matriz $(n \times m)$, crie uma matriz $(n + 2 \times m + 2)$ e utilize somente as posições (i, j), com $1 \leq i \leq n$ e $1 \leq j \leq m$, para trabalhar com o tabuleiro. Note que desse modo há de fato uma "moldura" em volta do tabuleiro que não será utilizada. Esta dica facilita as verificações das posições adjacentes: com a moldura, o código Java que verifica as posições adjacentes de uma posição numa fronteira do tabuleiro pode ser o mesmo que verifica as posições adjacentes a qualquer outra posição. Não será necessário verificar se os índices assumirão valores fora da matriz.*

Veja os exemplos de representação de um tabuleiro 8×8 com 10 minas.

	0	1	2	3	4	5	6	7	8	9
0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
2	-1	-1	**-2**	**-2**	-1	-1	-1	-1	**-2**	-1
3	-1	-1	-1	**-2**	-1	-1	**-2**	-1	-1	-1
4	-1	-1	-1	-1	-1	-1	**-2**	-1	-1	-1
5	-1	-1	**-2**	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	**-2**	-1	-1	-1	-1
7	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
8	-1	**-2**	-1	-1	-1	-1	-1	**-2**	-1	-1
9	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

(a) No começo do jogo

	0	1	2	3	4	5	6	7	8	9
0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
2	-1	-1	**-2**	**-2**	-1	-1	-1	-1	**-2**	-1
3	-1	-1	-1	**-2**	-1	-1	**-2**	3	1	-1
4	-1	-1	-1	-1	-1	-1	**-2**	2	0	-1
5	-1	-1	**-2**	-1	-1	-1	2	1	0	-1
6	-1	-1	-1	-1	-1	**-2**	1	**0**	0	-1
7	-1	-1	-1	-1	-1	-1	2	1	1	-1
8	-1	**-2**	-1	-1	-1	-1	-1	**-2**	-1	-1
9	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

(b) Depois de revelar a posição $(6, 7)$

Na construção do tabuleiro, será preciso gerar números aleatórios para se determinar as posições das minas. Utilize a classe `java.util.Random` para isso. Ela define o método `int nextInt(int n)` que devolve um inteiro pseudo-aleatório entre 0 (inclusive) e `n` (exclusive). Veja um exemplo de uso:

```
void testaRandom() {
  java.util.Random gerador = new java.util.Random();
  int i;
  System.out.println("Seqüência de números pseudo-aleatórios " +
                     "entre 0 e 99:");
  for (i = 0; i < 10; i++) {
    int x = gerador.nextInt(100);
    System.out.print(x + " ");
  }
  System.out.println();
}
```

Dica 4 *Para a depuração, pode ser inconveniente o fato do seu programa ter um certo comportamento não determinístico. É possível, porém, reproduzir a seqüência de números gerados pelo* `Random` *em execuções distintas do programa. Basta usar o construtor* `Random(int semente)`*. Para um mesmo valor de* `semente`*, o objeto* `Random` *criado devolverá sempre a mesma seqüência de números.*

Lembrando, é obrigatório que o primeiro palpite do jogador sempre seja uma posição livre. Caso não seja, é necessário transferir a mina na posição para outro lugar.

3.1.2 Exemplo de uso

Usando o interpretador do DrJava, podemos usar a classe `Tabuleiro` para jogar uma partida de Campo Minado (apesar dessa não ser uma maneira muito prática). Veja um exemplo.

```
> Tabuleiro tab = new Tabuleiro(1);
> tab.jogada(1,7)
1
> tab.imprimeTabuleiro()
    1 2 3 4 5 6 7 8
  /-----------------\
1 | + + 1 0 0 0 0 0 | 1
2 | + + 3 2 1 1 0 0 | 2
3 | + + + + + 1 0 0 | 3
4 | + + 3 2 1 1 0 0 | 4
5 | + + 1 0 0 0 1 1 | 5
6 | + + 3 2 1 1 2 + | 6
7 | + + + + + + + + | 7
8 | + + + + + + + + | 8
  \-----------------/
    1 2 3 4 5 6 7 8
```

```
> tab.jogada(3,5)
-1
> tab.imprimeTabuleiro()
    1 2 3 4 5 6 7 8
  /-----------------\
1 | @ @ 1 0 0 0 0 0 | 1
2 | + + 3 2 1 1 0 0 | 2
3 | + @ @ + @ 1 0 0 | 3
4 | + + 3 2 1 1 0 0 | 4
5 | + @ 1 0 0 0 1 1 | 5
6 | + + 3 2 1 1 2 @ | 6
7 | + + @ @ + + @ + | 7
8 | + + + + + + + + | 8
  \-----------------/
    1 2 3 4 5 6 7 8
```

3.2 JogadorHumano

Usando a classe `Tabuleiro`, a classe `JogadorHumano` deve permitir que uma pessoa jogue o Campo Minado usando o teclado. A leitura do teclado pode ser feita pela classe `SavitchIn`.

Cada objeto da classe `JogadorHumano` deve oferecer o método `public void iniciaJogo()`, que conduz uma partida de Campo Minado. Este método deve pedir que o jogador digite as informações necessárias no teclado.

Inicialmente, o jogador deve escolher entre os diversos níveis de dificuldade, ou um tabuleiro personalizado. Utilize opções numeradas (menus) e leitura de inteiros para essa parte.

Em seguida, o método entra num laço onde se pergunta as coordenadas da posição que o jogador deseja revelar. Este laço acaba quando o jogador ganha ou perde o jogo. Em cada iteração, devem ser impressas as seguintes mensagens, conforme a situação.

1. Posição livre descoberta. Além da mensagem, deve-se imprimir o novo estado do tabuleiro. Caso a posição descoberta seja a última, deve-se imprimir também uma mensagem de felicitação.
2. Posição livre já conhecida.
3. Posição minada. Neste caso, deve-se imprimir a posição de todas as minas.

Você também precisará implementar o método `public static void main(String args[])` para poder ler do teclado.

3.3 JogadorAutomatico

Esta classe utilizará a classe `Tabuleiro` para permitir que um programa de computador jogue o Campo Minado. Ela deve fornecer o método `public boolean iniciaJogo(Tabuleiro t)`. Esse método joga a partida de Campo Minado oferecida pelo tabuleiro `t`, devolvendo `true` se e somente se ele conseguiu vencer a partida.

Implemente a estratégia que quiser. Ela não precisa ser complicada: escolher posições aleatoriamente até o jogo terminar é perfeitamente aceitável para este EP.

Deve ser possível também para uma pessoa saber o que ocorreu na partida, então imprima mensagens semelhantes às impressas na classe `JogadorHumano` (talvez não seja conveniente imprimir o tabuleiro sempre).

4 Itens Opcionais

Com o objetivo de incrementar o seu programa, você pode:

- marcar o tempo do jogo;
- criar mais um método no `Tabuleiro` para permitir que posições suspeitas sejam marcadas, o que ajuda na resolução dos problemas (o mesmo que o clique com o botão direito do *mouse* na implementação para Windows);
- fazer com que o jogador automático seja mais inteligente, isto é, que observe as condições do tabuleiro, ao invés de jogar aleatoriamente. Para isto, será necessário adicionar um novo método na classe `Tabuleiro` que forneça uma matriz com as informações já obtidas.
- Criar uma interface gráfica.

Estes itens adicionais **não** serão considerados na correção.

5 Observações importantes

Sobre a elaboração:

- Este EP pode ser elaborado por equipes de dois alunos, contanto que se pratique exclusivamente a programação pareada. Lembrando, as regras são as seguintes.
  - Os alunos devem trabalhar sempre juntos, a idéia é que deve existir uma cooperação.
  - Mesmo a digitação do EP deve ser feita em grupo, enquanto um digita, o outro fica acompanhando o trabalho.
  - Caso em um grupo exista um aluno com maior facilidade, este deve explicar as decisões tomadas. E o seu par deve participar e se esforçar para entender o desenvolvimento do programa.

  Recomendamos fortemente que o exercício seja desenvolvido da forma descrita na observação acima. Se não for possível seguir todas as regras à risca, recomendamos que o EP seja feito individualmente.

- Escreva cada classe em um arquivo `.java` diferente. O nome de cada arquivo **deve** ser <*nome da classe*>`.java`. Isto é, `Tabuleiro.java`, `JogadorHumano.java`, e assim por diante.

  ATENÇÃO: veja a observação sobre a entrega pelo Panda.

Sobre a avaliação:

- No caso de exercícios feitos em dupla, a mesma nota da correção será atribuída aos dois alunos do grupo.
- Não serão toleradas cópias! Exercícios copiados (com ou sem eventuais disfarces) receberão nota ZERO (inclusive o original).
- Exercícios atrasados não serão aceitos.
- Exercícios com erros de compilação receberão nota ZERO. Para evitar problemas, verifique se o seu programa compila antes de entregá-lo.
- É muito importante que seu programa tenha comentários relevantes e esteja bem indentado, ou seja, digitado de maneira a ressaltar a estrutura de subordinação dos comandos do programa (conforme visto em aula). A qualidade do seu trabalho sob esse ponto de vista influenciará sua nota!
- As informações impressas pelo seu programa na tela devem aparecer da forma mais clara possível. Este aspecto também será levado em consideração no cálculo da sua nota.
- Uma regra básica é a seguinte: do ponto de vista do monitor responsável pela correção dos trabalhos, quanto mais convenientemente apresentado estiver o seu programa, melhor será a disposição dele para dar-lhe uma nota generosa.

Sobre a entrega:

- O prazo de entrega é o dia 27/6/2003.

- No início de cada arquivo, acrescente um cabeçalho bem informativo, semelhante ao seguinte:

```
/*************************************************************/
/**   MAC 110 - Introdução à Computação                     **/
/**   IME-USP - Primeiro Semestre de 2003                   **/
/**   <turma> - <nome do professor>                         **/
/**                                                         **/
/**   Terceiro Exercício-Programa  --  Campo Minado         **/
/**   Arquivo: <nome do arquivo>                            **/
/**                                                         **/
/**   <nome do(a) aluno(a)>                <número USP>     **/
/**   <nome do(a) aluno(a)>                <número USP>     **/
/**                                                         **/
/**   <data de entrega>                                     **/
/*************************************************************/
```

- Para a entrega, utilize o Panda. Para isso, o seu EP deve estar todo contido num único arquivo. Como este EP consiste de vários, utilize um empacotador para juntar todos os arquivos num só. Empacote apenas as classes que você escreveu. No Windows, você pode usar o programa WinZip (ou equivalente) e entregar o arquivo `.zip` gerado. No Linux, você pode usar o comando *tar* da seguinte forma.

  ```
  $ cd <diretório do seu EP>
  $ tar zcvf <arquivo empacotado> <arquivos do seu EP>
  ```

  Por exemplo:

  ```
  $ cd ep3
  $ tar zcvf ep3.tgz *.java
  ```

  Nesse caso, o que deve ser entregue é o arquivo `ep3.tgz`. Para mais detalhes sobre o comando *tar*, execute o comando `man tar`.

  Para evitar problemas, **não deixe de tirar as suas dúvidas** a respeito desse procedimento com antecedência.

  Você pode entregar várias versões de um mesmo EP até o prazo, mas somente a última será armazenada pelo sistema. Encerrado o prazo, o sistema não aceitará mais os EPs. Os procedimentos de entrega para trabalhos individuais e em grupo diferem um pouco, consulte a ajuda do Panda para obter mais detalhes.

- Guarde uma cópia do seu EP pelo menos até o fim do semestre!

Este documento foi feito em LaTeX.