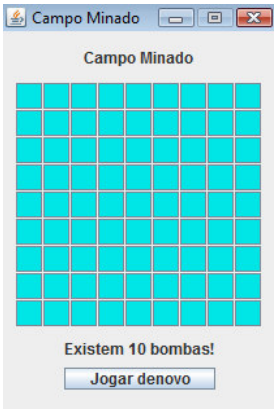
Trabalho B – Campo Minado

Neste trabalho, você deve implementar um jogo chamado Campo Minado. Neste documento, você encontrará todas as informações sobre a implementação a ser realizada: regras do jogo, classes a serem implementadas, classes disponibilizadas, data de entrega e arquivos para serem entregues.

1. Regras do Jogo Campo Minado

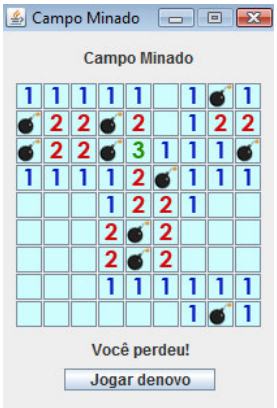
O Campo Minado é um jogo aparentemente simples de memória e raciocínio. O objetivo do Campo Minado é revelar os quadrados vazios e com números, evitando aqueles que escondem bombas. Quantidade de bombas no cenário: 10.

Tela inicial do jogo:



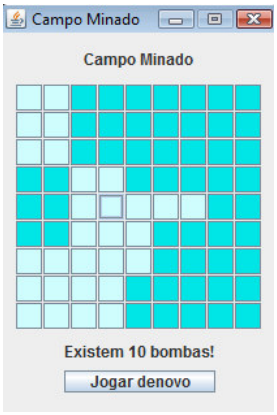
- Você pode revelar o que existe em um quadrado clicando nele. Se você revelar uma bomba, perderá o jogo.

- O número que aparece no quadrado indica o número total de bombas nos oito quadrados que o cercam. Você pode usar esse número para ajudá-lo a deduzir se é seguro revelar um quadrado. Quadrados vazios indicam que não há bombas nos 8 quadrados que o cercam. Uma situação exemplo pode ser verificada na figura abaixo:

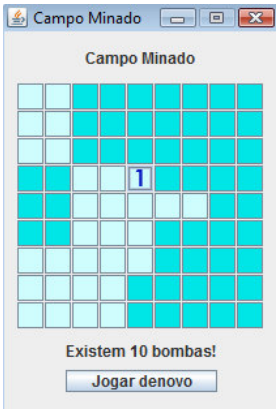


Conforme você vai revelando os quadrados, a tela vai apresentando o resultado da sua escolha. Neste sentido, 3 situações podem acontecer:

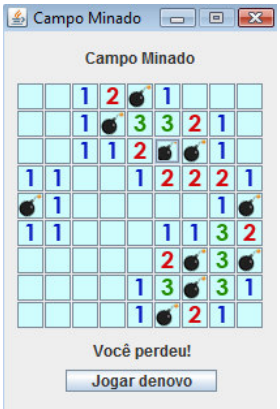
a. **você clicou em um espaço em branco:** todos os espaços em branco em volta são automaticamente revelados, conforme ilustra a figura abaixo, simulando que a posição marcada foi clicada.



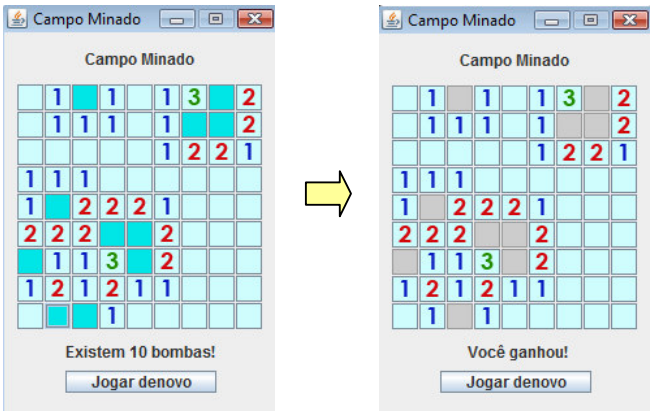
b. **você clicou em um número:** o número é revelado, conforme pode ser visto abaixo:



c. **você clicou em uma bomba:** todas as posições são reveladas e você perdeu o jogo, conforme a figura abaixo:



Você ganha o jogo quando você revelar **todas as posições que não são bombas** (ou seja, quando restarem apenas os 10 quadrados que contêm as 10 bombas). A figura abaixo ilustra um exemplo de uma penúltima e última jogada antes de ganhar o jogo, respectivamente.

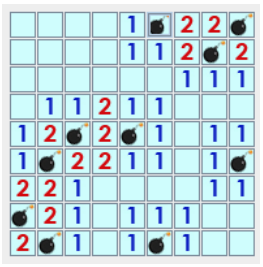


2. O Trabalho

Antes de mais nada: faça o download na página da disciplina do arquivo *CampoMinado.zip*. Este arquivo trata-se de um *template* (modelo) de projeto do BlueJ, que já contém algumas classes (que serão explicadas no decorrer deste documento).

Você deverá imaginar a tela do jogo como uma matriz (array bidimensional) de dimensões **9x9**. Esta matriz **deve** ser uma matriz de **inteiros**, na qual as posições em **0** representam posições vazias, as posições em **-1** representam os locais com bombas e as posições com **números** representam o número de bombas. Por exemplo, a matriz abaixo refere-se à tela ao lado:

```
 0  0  0  0  1 -1  2  2 -1
 0  0  0  0  1  1  2 -1  2
 0  0  0  0  0  0  1  1  1
 0  1  1  2  1  1  0  0  0
 1  2 -1  2 -1  1  0  1  1
 1 -1  2  2  1  1  0  1 -1
 2  2  1  0  0  0  0  1  1
-1  2  1  0  1  1  1  0  0
 2 -1  1  0  1 -1  1  0  0
```



O seu trabalho terá 3 classes: *Interface*, *CampoMinado* e *Main*. Você deverá implementar as classes **CampoMinado** e **Main**. A classe **Interface** está disponibilizada para o uso dentro do projeto *template* na página da disciplina.

2.1 Classe CampoMinado

No *template* do projeto do BlueJ disponibilizado, encontra-se um *template* desta classe (*CampoMinado.java*). Você deverá utilizá-lo como ponto de partida. Esta classe possui os seguintes atributos:

```
private Interface inter;
private int[][] tela;
```

Onde `inter` é uma referência para um objeto do tipo Interface, que apresentará a interface gráfica na tela e `tela` é um array bidimensional (matriz) de inteiros que representa a tela do jogo (que deve ser de dimensões 9x9).

Esta classe possui os seguintes métodos:

- `public CampoMinado()`: construtor da classe CampoMinado, que deve inicializar os atributos `inter` e `tela`. A inicialização de `inter` já está pronta no template, você deve inicializar a matriz `tela` (inicialize a matriz tela com todas as posições representando quadrados vazios no jogo).

- `public void inicializaTela()`: método que distribui as 10 bombas na matriz `tela`, indicando com valor -1 (que representa uma bomba) a posição escolhida. A escolha das posições dar-se-á através da utilização da instrução `(int)(Math.random() * 9)` para a escolha da linha e `(int)(Math.random() * 9)` para a escolha da coluna.

ATENÇÃO: não esqueça de testar se na posição que você vai colocar a bomba já não existe uma outra bomba. Ao final, 10 bombas devem estar na matriz `tela`.

Após a distribuição das bombas, você deve verificar para cada posição da matriz que não for uma bomba, quantas bombas existem na sua volta (nos quadrados em volta daquela posição da matriz). As posições possuem 8 quadrados em volta, porém, note que isto não é verdade para as posições das bordas da matriz. **Você deve tratar isto**.

Quando este método acabar, a matriz `tela` estará assim:

- **posições em 0:** representam um espaço em branco
- **posições em -1:** representam uma bomba
- **posições com números:** representam a quantidade de bombas (valores em -1) em volta daquela posição

- `public void imprimeTela()`: este método imprime na tela a matriz `tela`, no mesmo formato do jogo, conforme exemplo abaixo:

```
 1  1  2 -1 -1  1  0  0  0
 1 -1  2  2  2  1  0  0  0
 2  2  1  0  1  1  1  0  0
-1  1  0  0  1 -1  1  0  0
 1  1  0  0  1  1  1  0  0
 0  0  0  0  0  0  1  1  1
 0  0  1  1  1  0  1 -1  1
 1  2  4 -1  2  0  1  1  1
 1 -1 -1 -1  2  0  0  0  0
```

Note que este método serve para que você saiba se estão funcionando corretamente as posições escolhidas, podendo até mesmo comparar ao que está acontecendo com a interface gráfica.

- **`public void reinicia()`**: que reinicia o jogo para caso o jogador queira jogar novamente ou apenas recomeçar o jogo. Neste método, resta apenas a reinicialização da matriz `tela`, que deve ter seus valores todos reiniciados em 0, indicando posições vazias no jogo (pois o jogo irá começar novamente).

- **`public void posicaoEscolhida(int linha, int coluna)`**: este método é chamado automaticamente pela interface quando o jogador clicar em um quadrado no jogo. Os parâmetros `linha` e `coluna` representam exatamente as posições que o jogador clicou na tela. Logo, o que você deve fazer neste método é o seguinte:
    a. se a posição escolhida pelo jogador for uma bomba, você deve acionar o método `clicouBomba` do atributo `inter`, passando por parâmetro a matriz `tela`;

    b. se a posição escolhida pelo jogador for um vazio, você deve acionar o método `clicouVazio` do atributo `inter`, passando por parâmetro (nesta ordem) a `linha` escolhida, a `coluna` escolhida e a matriz `tela`;

    c. se a posição escolhida pelo jogador não for nem uma bomba e nem um vazio, é porque trata-se de um número. Então, você deve acionar o método `clicouNumero` do atributo `inter`, passando por parâmetro (nesta ordem) a `linha` escolhida, a `coluna` escolhida e o número que o usuário clicou.

Note que você poderá criar quantos métodos auxiliares você quiser/precisar. Porém, estes métodos já encontram-se no template, e devem ser utilizados para realizar o que foi descrito.

2.2 Classe Interface

A classe Interface já foi implementada, e possui métodos públicos e privados. Você não deve alterar a classe Interface em nada. Note que o método que faz com que os espaços vazios adjacentes sejam abertos automaticamente (`procuraVazios`) encontra-se nesta classe, não sendo necessária a implementação deste no trabalho.

Os métodos que você vai chamar a partir da sua classe são: `clicouBomba`, `clicouVazio` e `clicouNumero`. Estes métodos vão interagir com a interface gráfica, desenhando o que foi escolhido na tela.

Nesta classe também está implementado o final do jogo, tanto quando o jogador ganha quanto quando ele perde. Logo, o controle do final do jogo também não é necessário implementar.

2.3 Classe Main

Esta classe apenas possui o método `main`. Neste método, declare e inicialize um objeto do tipo CampoMinado. E pronto. Sim, é só isto mesmo, uma linha de código.

3. Informações gerais

- **Não mexa nos arquivos de imagens** e demais arquivos que encontram-se dentro do projeto disponibilizado na página.

- O trabalho poderá ser realizado individualmente ou (no máximo) em duplas

- Você deverá entregar:
  - o **projeto compactado** (ZIP) do BlueJ com o nome: *Seu_Nome_TrabalhoB.zip*.
  - um **relatório** com no máximo **2 páginas**, no formato disponibilizado na página da disciplina (faça o download do arquivo: *Template_Relatorio_TrabalhoB.doc*). A descrição do que escrever no relatório encontra-se no próprio documento.
  - a entrega deve ser realizada via-email: **mraeder@unisinos.br** com o assunto *TRABALHO B – SEU NOME – SUA TURMA*. Identifique-se claramente no e-mail.

- Data de entrega: **IMPRETERIVELMENTE** até as 23h59min do dia 21 de junho de 2009

4. Avaliação

Este trabalho vale **30% da nota do GrauB**. A avaliação será dada conforme os critérios que seguem:

- Código comentado: 0,5 pontos
- Código identado: 0,5 pontos
- Implementação correta do programa: 7,5 pontos
- Relatório: 1,5 pontos

5. Dicas

- Não ache que o trabalho é impossível antes de começar. Você é totalmente capaz de faze-lo.
- Faça uma implementação simples porém correta. Não é necessário inventar coisas que complicam o código. Invenções não ganham nota extra.
- Faça passo a passo. Não faça tudo e depois verifique se funciona. Coloque as bombas, e certifique-se de que estão corretas. Coloque os números e certifique-se que estão corretos.
- Se quiser testar sem integrar com a Interface, crie uma classe auxiliar, apenas com a matriz. Depois que a matriz estiver correta, com as bombas e os números inicializados corretamente, a parte mais fácil é integrar com a interface.
- Não deixe para fazer no último momento. Você tem pouca coisa para implementar, mas tem bastante coisa para pensar.
- Reserve alguma parte do dia para pensar sobre o trabalho e colocar em prática algumas idéias.
- Qualquer dúvida, mande por e-mail, que serão respondidas na medida do possível.

Bom trabalho!