Trabalho: Genius e Tabela de Recordes

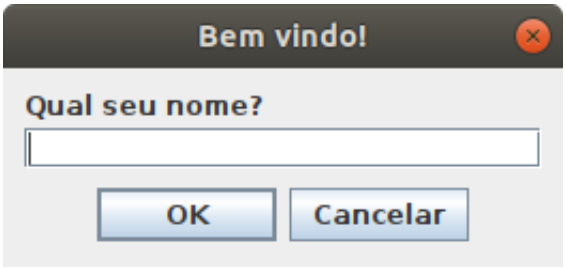
Muito popular na década de 1980, o brinquedo Genius buscava estimular a memorização de cores. Com um formato semelhante a um OVNI, possuía botões coloridos que emitiam sons harmônicos e se iluminavam em sequência. Cabia aos jogadores repetir a ordem das cores sem errar.

Figura 1: Jogo Genius.

Ao invés de cores, utilizaremos números. A cada iteração, nosso programa irá sortear um algarismo aleatório, que será exibido para o usuário. Caberá ao usuário memorizar e digitar corretamente toda a sequência de caracteres exibidos.

O programa irá interagir com o usuário através de janelas de diálogo. Para abrir uma janela deste tipo, utilize a classe **JOptionPane**, que permite exibir caixas de diálogo de forma simplificada. Neste jogo, utilizaremos 3 métodos estáticos fornecidos por esta classe para exibição de janelas:

1. **showInputDialog**: Pede que o usuário digite uma entrada. Por exemplo, para exibir a tela abaixo:

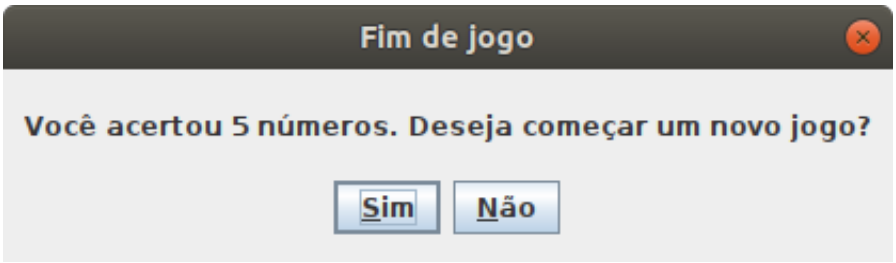


utilizamos o código a seguir:

```
String nome = JOptionPane.showInputDialog(null,"Qual seu nome?", "
    Bem vindo!", JOptionPane.PLAIN_MESSAGE);
```

Se o usuário cancelar ou fechar a janela, a String retornada será *null*.

2. **showConfirmDialog**: Faz uma pergunta, e pede que o usuário confirme (Sim / Não / Cancelar). Por exemplo, para exibir a janela a seguir:

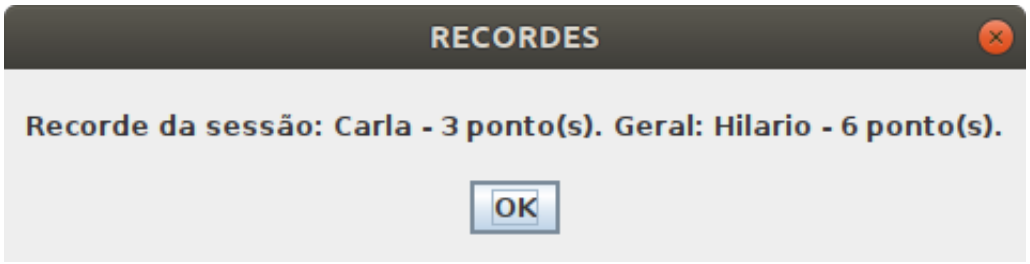


utilizamos o código a seguir:

```
int x = JOptionPane.showConfirmDialog(null,"Voce acertou 5 numeros.
    Deseja comecar um novo jogo?", "Fim de jogo", JOptionPane.
    YES_NO_OPTION);
```

Se o usuário escolher a opção "SIM", o retorno será 0. Se escolher "NÃO", o retorno será 1.

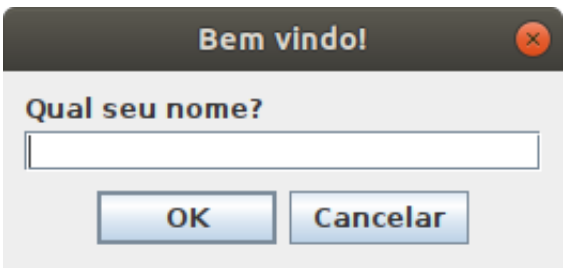
3. **showMessageDialog**: Informa o usuário sobre algo que occorreu. Por exemplo, para exibir a janela a seguir:



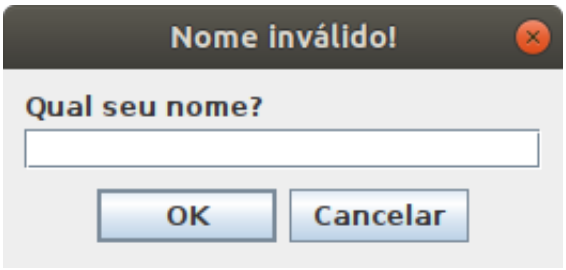
utilizamos o código a seguir:

```
JOptionPane.showMessageDialog(null, "Recorde da sessao: Carla = 3
    ponto(s). Geral: Hilario - 6 ponto(s).", "RECORDES", JOptionPane.
    PLAIN_MESSAGE);
```

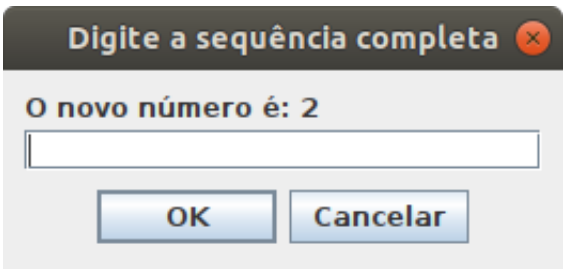
Ao entrar no programa, deve ser exibida uma janela de boas vindas, solicitando que o usuário digite seu nome:



Se o usuário digitar um nome vazio, cancelar ou fechar a janela, o programa deve insistir para que ele forneça seu nome:



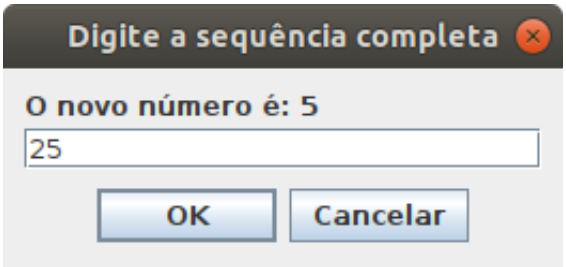
Após o usuário digitar seu nome, é exibida uma tela mostrando o novo número sorteado, e pedindo que ele digite toda a sequência sorteada até agora:



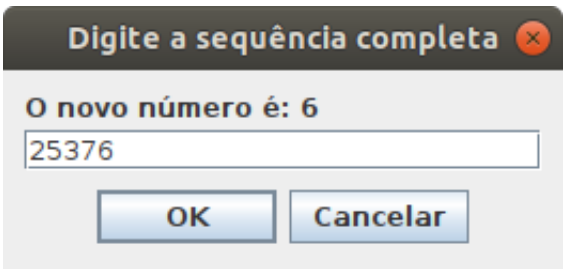
Para gerar um número aleatório, utilize a classe **Random**. O método **nextInt(x)** gera um número aleatório $0 \leq i < x$. Portanto, para gerar um número entre 1 e 9, podemos utilizar:

```
1 Random r = new Random();
2 int numero = r.nextInt(9) + 1;
```

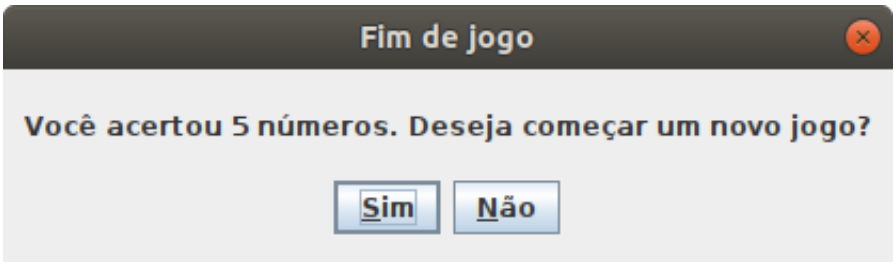
No exemplo da figura, ele deve digitar apenas o número 2 (o primeiro sorteado). Em seguida, supondo que o segundo número sorteado foi 5, ele deve digitar toda a sequência digitada até agora (ou seja, 25 – primeiro o 2 e depois o 5):



Dessa forma, após sortear os números 2, 5, 3, 7 e 6, ele deve digitar exatamente a sequência 25376:



Quando o usuário errar, o programa deve perguntar se ele quer começar a jogar de novo:



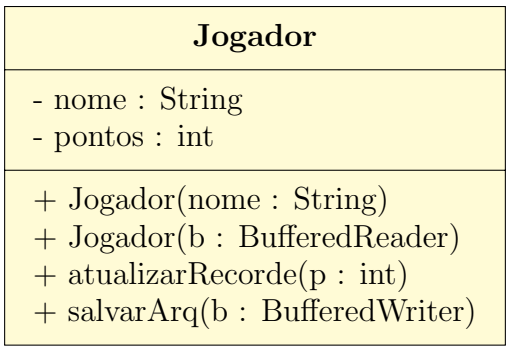
Se ele optar por jogar novamente, o nome do usuário será solicitado novamente e a pontuação atual será reiniciada.

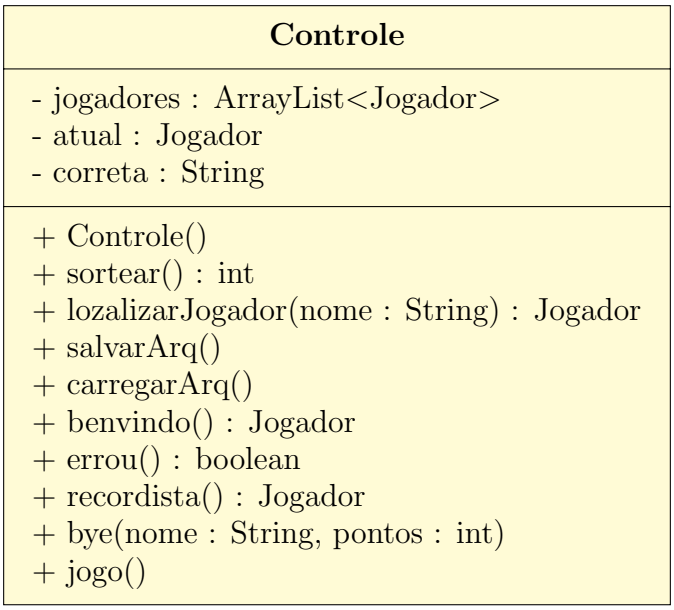
Ao final, quando ele errar e não quiser mais continuar jogando, o programa irá exibir uma mensagem com o recorde da sessão (usuário e pontuação máxima obtida desde que o programa foi aberto) e o recorde geral:

Para que haja um recorde geral, deve ser salvo um único arquivo texto contendo a pontuação máxima obtida por **todos** os usuários que já jogaram. O usuário será identificado apenas por seu nome. Se o usuário "Hilario" já estiver no arquivo com a pontuação 4, e em uma nova execução um outro usuário chamado "Hilario" fizer 6 pontos, sua entrada no arquivo será substituída.

Ao iniciar o programa, os dados da tabela de recorde serão carregados do arquivo os nomes e pontuações máximas dos usuários serão armazenados num ArrayList.

Seu programa terá duas classes, além da classe com o método principal:





- Classe **Jogador**: Armazena nome e pontuação máxima de cada usuário que já jogou. O primeiro construtor é chamado quando ainda não existe um usuário com aquele nome. Neste caso, sua pontuação máxima inicia com zero. O segundo construtor é chamado durante o carregamento do arquivo com os recordes.

  O método **atualizarRecorde** recebe uma pontuação como parâmetro e atualiza os pontos do jogador **caso** a pontuação recebida seja maior que a armazenada.

  O método **salvarArq** salva nome e pontuação do jogador no buffer recebido como parâmetro.

- Classe **Controle**:

  O construtor deve:

  - inicializar a String correta como vazia (“”), de jogadores
  - inicializar o ArrayList de jogadores e chamar o método que carrega os recordes do arquivo, que irá atualizar este ArrayList com os dados lidos do arquivo.
  - chamar o método de boas vindas, que irá retornar o usuário que está jogando no momento.

O método **sortear** gera o número aleatório entre 1 e 9, **adiciona-o** ao final da String correta, e retorna o número.

O método **localizarJogador** recebe o nome do jogador e busca por esse usuário no ArrayList de jogadores. Se ele existir, retorna-o. Caso contrário, cria um novo objeto para ele, insere-o no ArrayList, e retorna este novo objeto.

O método **salvarArq** salva num arquivo texto todas as infromações necessárias sobre o placar (quantidade de entradas no placar, nome e pontuação de cada jogador). O método deve cuidar do tratamento de erros e exceções necessários.

O método **carregarArq** faz a leitura do placar, atualizando o ArrayList de jogadores. O método deve cuidar do tratamento de erros e exceções necessários.

O método **benvindo** exibe a janela de boas vindas, solicitando o nome do jogador, e utiliza o método **localizarJogador** para retorna o jogador selecionado. O método deve cuidar de verificar se o nome não foi deixado em branco.

O método **errou** exibe a janela perguntando se o jogador deseja continuar jogando após errar a sequência, retornando verdadeiro ou falso.

O método **recordista** busca e retorna o usuário com a maior pontuação geral no placar.

O método **bye** exibe a janela final, mostrando o recorde da sessão (maior pontuação desde que o programa foi aberto) e o recorde geral.

O método **jogo** tem o loop responsável por ir sorteando novos números, e solicitando que o usuário digite a sequência, utilizando os métodos apropriados para isso.

- Classe Principal: A classe com o método principal cria o objeto da classe Controle e executa o método jogo enquanto o usuário optar por continuar jogando. Ao final, executa o método que exibe os placar final, passando como parâmetro o nome e pontuação do recordista da sessão (o recordista geral pode ser obtido do ArrayList, e não precisa ser passado como parâmetro).

Observações

- O trabalho vale 30 pontos.
- O código deve ser feito em Java.
- O trabalho pode ser feito em até 2 integrantes.
- Trabalhos entregues após o prazo serão automaticamente rejeitados.
- Trabalhos considerados plágio terão nota zero e serão enviados ao conselho de ética.

- Trabalhos com erro de execução, com formato de saída incorreto, ou que não compilarem terão nota 0.
- O trabalho deve ser enviado na sala da disciplina do AVA.
- Em caso de dúvidas na especificação do trabalho ou no próprio trabalho, contate-me em hsjunior@gmail.com