Exercícios 4.01 ao 4.10

Exercício 4.01 - Abra o projeto 'notebook1' no BlueJ e crie um objeto 'Notebook'. Armazene algumas notas nele - são strings simples -, então verifique se o número de notas retornado por 'numberOfNotes' corresponde ao número que você armazenou. Quando você utiliza o método 'showNotes', é necessário utilizar um valor de parâmetro de 0 (zero) para imprimir a primeira nota, 1 (um) para imprimir a segunda nota e assim por diante. Explicaremos a razão para essa numeração em um momento oportuno.
R: teste BlueJ

Exercício 4.02 - Implemente um método 'removeNote' em seu bloco de notas.
R:

```
public void removeNote(int numeroNota)
{
     if(numeroNota < 0)
     {
          //Esse número não é válido, pois é menor que zero.
     }
     else if(numeroNota < numberOfNotes())
     {
          //número válido para nota
          notes.remove(numeroNota);
     }
     else
     {
          //Esse número não é valido, pois é maior que o
     número de notas
     }
}
```

Exercício 4.03 - Com o que deveria se parecer o cabeçalho de um método 'listAllNotes'? Que tipo de retorno ele teve ter? Ele precisa de algum parâmetro?
R:

```
public void listAllNotes()
{

}
```

Não precisa de tipo de retorno, pois não retorna nenhum valor, apenas imprime. Não precisa de parâmetro, pois ele vai exibir todas as notas.

Exercício 4.04 - Sabemos que a primeira nota é armazenada no índice zero no 'ArrayList', então, também poderíamos escrever um corpo de 'listAllNotes' ao longo das linhas a seguir?

```
System.out.println(notes.get(0));
System.out.println(notes.get(1));
System.out.println(notes.get(2));
```

etc.
R: Não, pois o limite final de notas varia, e assim não é possível prever até qual nota irá ser impressa.

Exercício 4.05 - Implemente o método 'listNotes' em sua versão do projeto bloco de notas. (Uma solução com esse método implementado é fornecida na versão 'notebook2' desse projeto, mas, para melhorar seu conhecimento do assunto, recomendamos que você próprio escreva esse método.)
R:

```
public void listNotes()
{
     int i = 0;
     while (i < notes.size())
     {
          System.out.println(i + ": " + notes.get(i));
          i ++;
     }
}
```

Exercício 4.06 - Crie um 'Notebook' e armazene algumas notas nele. Utilize o método 'listNotes' para imprimí-las e verificar se o método funciona como deveria.
R: Funciona

Exercício 4.07 - Se desejar, você poderia utilizar o depurador para ajudá-lo a entender como as instruções no corpo do loop while são repetidas. Configure um ponto de interrupção um pouco antes do loop e passe pelo método até que a condição do loop avalie para 'false'.
R: Teste no BlueJ

Exercício 4.08 - Modifique 'showNote' e 'romeveNote' para imprimir uma mensagem de erro se o número da nota inserido não for válido.
R:

```
public void showNote(int noteNumber)
{
     if(noteNumber < 0)
     {
          System.out.println("Esse número não é válido, pois
     é menor que zero.");
     }
     else if(noteNumber < numberOfNotes())
     {
          System.out.println(notes.get(noteNumber));
     }
     else
     {
          System.out.println("Esse número não é valido, pois
     é maior que o número de notas.");
     }
}

public void removeNote(int numeroNota)
{
     if(numeroNota < 0)
     {
          System.out.println("Esse número não é válido, pois
     é menor que zero.");
     }
     else if(numeroNota < numberOfNotes())
     {
          notes.remove(numeroNota);
     }
     else
     {
          System.out.println("Esse número não é valido, pois
     é maior que o número de notas.");
     }
}
```

Exercício 4.09 - Modifíque o método 'listNotes' de modo que imprima o valor da variável local 'index' na frente de cada nota. Por exemplo:
0: Buy some bread.
1: Recharge phone.
2: 11.30: Meeting with John.
Isso facilita muito o funcionamento do índice correto ao remover uma nota.
R:

```
public void listNotes()
{
     int i = 0;
     while (i < notes.size())
     {
          System.out.println(i + ": " + notes.get(i));
     i ++;
     }
}
```

Exercício 4.10 - Em uma única execução do método 'listNotes', a coleção de notas é questionada repetidamente sobre quantas notas estão armazenadas atualmente. Isso é feito toda vez que a condição de loop é verificada. O valor retornado por tamanho varia de uma verificação para a próxima? Se você acha que a resposta é "Não", então reescreva o método 'listNotes' de modo que o tamanho da coleção de notas seja determinado somente uma vez e armazenado em uma variável local antes da execução do loop. Então utilize a variável local na condição do loop em vez da chamada 'size'. Verifique se essa versão fornece os mesmos resultados. Se você tiver problemas em concluir esse exercício, tente utilizar o depurador para saber onde está o erro.
R: O valor do tamanho do ArrayList não muda de uma verificação para a outra, pois ele não é modificado dentro do loop. A nova versão do 'listNotes' fornece o mesmo resultado.

```
public void listAllNotes()
{
     int i = 0;
     int size = notes.size();
     while (i < size)
     {
          System.out.println(i + ": " + notes.get(i));
          i ++;
     }
}
```