Instruções para a entrega: fazer os exercícios a seguir e entregar as respostas em papel para o professor no início da aula do dia 15/mar. A entrega pode ser em dupla, alunos ausentes não terão a nota considerada.

Considere as classes a seguir nos exercícios.

```
public class No {
  int conteudo;
  No proximo;
}
```

```
public class Lista {
   No inicio;

   No getInicio() {
      return inicio;
   }

   void add(No no) {
      if( !exists(no.conteudo) ) {
         if( inicio == null ) {
            inicio = no;
         }
         else {
            No aux = inicio;
            while( aux.proximo != null ) {
               aux = aux.proximo;
            }
            aux.proximo = no;
         }
      }
   }

   boolean exists(int nro) {
      No no = inicio;
      while( no != null ) {
         if( no.conteudo == nro ) {
            return true;
         }
         no = no.proximo;
      }
      return false;
   }

   void print() {
      No no = inicio;
      int cont = 0;
      while( no != null ) {
         System.out.println(
               ++cont + " - " + no.conteudo );
         no = no.proximo;
      }
   }

   void interrupt(int nro) {
      //programar aqui o corpo do método
   }
}
```

Exercício 1 – O que é necessário alterar no método ***print*** para ele entrar em loop infinito?

Exercício 2 – O trecho de código a seguir se encontra no método ***add***. Por que o código passa a apresentar erro se mudarmos a comparação de igualdade (==) para diferente (!=)?

```
if( inicio == null ) {
   inicio = no;
}
```

Exercício 3 – O trecho de código a seguir se encontra no método ***add***. Por que após fazer a mudança a seguir a lista mantém apenas o 1º valor adicionado?

```
else {
   No aux = inicio;
   while( aux.proximo != null ) {
      aux = aux.proximo;
   }
   aux.proximo = no;
}
```

Mudar para:

```
else {
   No aux = inicio;
   while( aux.proximo != null ) {
      aux = aux.proximo;
      aux.proximo = no;
   }

}
```

Exercício 4 – Completar o código a seguir para unir as duas listas, ou seja, encadear uma lista na outra.

Observação: não é permitido alterar o código da classe Lista.

```
public class Principal {
   public static void main(String[] args) {
      Lista a = new Lista(), b = new Lista();
      No no;
      no = new No();
      no.conteudo = 9;
      a.add(no);
      no = new No();
      no.conteudo = 5;
      a.add(no);
      no = new No();
      no.conteudo = 2;
      a.add(no);
      no = new No();
      no.conteudo = 4;
      a.add(no);
      no = new No();
      no.conteudo = 5;
```

```
        a.add(no);


        no = new No();
        no.conteudo = 20;
        b.add(no);
        no = new No();
        no.conteudo = 10;
        b.add(no);
        no = new No();
        no.conteudo = 15;
        b.add(no);
        //código para unir as duas listas

        _____________________________

        _____________________________

        _____________________________
    }
}
```

Exercício 5 – Considere que uma lista possui os seguintes elementos nesta ordem:

```
5, 9, 11, 4, 2, 3, 8, 1
```

Ao chamar o método `interrupt(4)` a lista passará a ter somente os elementos:

```
5, 9, 11, 4
```

O método ***interrupt*** interrompe a lista no elemento que tiver o conteúdo passado como parâmetro.

Programar o corpo do método ***interrupt*** na classe ***Lista***.