

# Relembrando...

## Tarefas

1) Modifique o projeto no qual estamos trabalhando para que seja escrito em um arquivo de *log* (log.txt) sempre que ocorrer um erro no programa. Ou seja, sempre que entrar em um *catch*.

# Relembrando...

```
1  /*
2   * Classe auxiliar com funções para uso de arquivos.
3   */
4  package aula9exemplos;
5
6  import java.io.BufferedWriter;
7  import java.io.File;
8  import java.io.FileInputStream;
9  import java.io.FileOutputStream;
10 import java.io.FileWriter;
11 import java.io.IOException;
12 import java.io.ObjectInputStream;
13 import java.io.ObjectOutputStream;
14 import java.io.PrintWriter;
15 import java.util.Calendar;
16
17 /**
18  *
19  * @author coelho
20  */
21 public class Arquivo {
22
```

# Relembrando...

```
22
23     private static final String ARQUIVO_LOG = "log.txt";
24     private static final String ARQUIVO_BINARIO_ALUNOS = "alunos.bin";
25     private static final String ARQUIVO_BINARIO_ENSINO = "ensino.bin";
26
27     private static final int TOTAL_ALUNOS = 1000;
28
29     private static final String DIRETOR_ENSINO = "Pâmela Perini";
30     private static final String COORDENADOR_ENSINO = "Vitor Valente";
31
```

# Relembrando...

```
32 public static boolean escreve_log(String texto) {  
33     try {  
34         File f = new File(ARQUIVO_LOG);  
35         if (!f.exists()) {  
36             f.createNewFile();  
37         }  
38         try (FileWriter fw = new FileWriter(f, true);  
39             BufferedWriter bw = new BufferedWriter(fw);  
40             PrintWriter pw = new PrintWriter(bw)) {  
41             String hora_data_atual = String.valueOf(Calendar.getInstance().getTime());  
42  
43             pw.print(hora_data_atual + ": ");  
44             pw.println(texto);  
45             System.err.println("LOG: " + texto);  
46         }  
47     } catch (IOException e) {  
48         System.err.println("Erro ao criar o arquivo " + ARQUIVO_LOG);  
49         return false;  
50     }  
51     return true;  
52 }  
53
```

# Relembrando...

```
47 try {
48     opcao = menu("MENU 1: \n "
49                 + "[" + OP_ALUNO + "] Aluno \n "
50                 + "[" + OP_PROFESSOR + "] Professor \n "
51                 + "[" + OP_ENSINO + "] Setor de Ensino \n "
52                 + "[" + OP_SAIR + "] Sair \n",
53                 br);
54 } catch (IOException e) {
55     Arquivo.escreve_log("Erro ao ler opção do menu 1");
56 }
```

```
57 switch (opcao) {
58     case OP_ALUNO:
59         try {
60             menu_alunos("MENU 2: \n "
61                         + "[" + OP_ALUNO_VER_CURSOS + "] Ver Cursos \n"
62                         + "[" + OP_ALUNO_VER_NOTAS + "] Ver notas \n"
63                         + "[" + OP_ALUNO_VOLTAR + "] Voltar",
64                         ensino,
65                         alunos,
66                         br);
67         } catch (IOException e) {
68             Arquivo.escreve_log("Erro ao ler opção do menu do aluno");
69         }
70         break;
```

# Relembrando...

## Tarefas

2) Modifique o programa principal para que sejam abertos no início e salvos no final de cada execução os dados acadêmicos. Para isto, precisam ser salvos a lista de alunos (dados do *array* alunos) e os dados do setor de ensino (encapsulados no objeto ensino). Use um arquivo binário chamado ifrs.bin para armazenar os dados dos objetos.

# Relembrando...

```
54 public static boolean salva_ensino(SetorEnsino ensino) {  
55     try {  
56         File f = new File(ARQUIVO_BINARIO_ENSINO);  
57         if (!f.exists()) {  
58             f.createNewFile();  
59         }  
60         try (FileOutputStream fos = new FileOutputStream(f);  
61             ObjectOutputStream oos = new ObjectOutputStream(fos)) {  
62             oos.writeObject(ensino);  
63         }  
64         System.out.println("Dados salvos com sucesso: ");  
65         System.out.println(ensino.toString());  
66     } catch (IOException e) {  
67         Arquivo.escreve_log("Erro ao criar o arquivo " + ARQUIVO_BINARIO_ENSINO);  
68     }  
69     return true;  
70 }  
71
```

# Relembrando...

```
public static boolean salva_alunos(Aluno[] alunos) {  
    try {  
        File f = new File(ARQUIVO_BINARIO_ALUNOS);  
        if (!f.exists()) {  
            f.createNewFile();  
        }  
        try (FileOutputStream fos = new FileOutputStream(f);  
             ObjectOutputStream oos = new ObjectOutputStream(fos)) {  
            oos.writeObject(alunos);  
        }  
        System.out.println("Dados salvos com sucesso: ");  
        for (Aluno aluno : alunos) {  
            System.out.println(aluno.toString());  
        }  
    } catch (IOException e) {  
        Arquivo.escreve_log("Erro ao criar o arquivo " + ARQUIVO_BINARIO_ALUNOS);  
    }  
    return true;  
}
```



# Relembrando...

```
112 public static SetorEnsino obtem_ensino() {  
113     SetorEnsino ensino = new SetorEnsino(DIRETOR_ENSINO, COORDENADOR_ENSINO);  
114     try {  
115         File f = new File(ARQUIVO_BINARIO_ENSINO);  
116         if (!f.exists()) {  
117             f.createNewFile();  
118         }  
119         try (FileInputStream fis = new FileInputStream(f);  
120             ObjectInputStream ois = new ObjectInputStream(fis)) {  
121             ensino = (SetorEnsino) ois.readObject();  
122         } catch (ClassNotFoundException ex) {  
123             Arquivo.escreve_log(  
124                 "Erro ao ler dados do setor de ensino salvos no arquivo "  
125                 + ARQUIVO_BINARIO_ENSINO);  
126         }  
127     } catch (IOException e) {  
128         Arquivo.escreve_log("Erro ao criar o arquivo " + ARQUIVO_BINARIO_ENSINO);  
129     }  
130     return ensino;  
131 }  
132 }
```

# Relembrando...

```
92 public static Aluno[] obtem_alunos() {  
93     Aluno[] alunos = new Aluno[TOTAL_ALUNOS];  
94     try {  
95         File f = new File(ARQUIVO_BINARIO_ALUNOS);  
96         if (!f.exists()) {  
97             f.createNewFile();  
98         }  
99         try (FileInputStream fis = new FileInputStream(f);  
100             ObjectInputStream ois = new ObjectInputStream(fis)) {  
101             alunos = (Aluno[]) ois.readObject();  
102         } catch (ClassNotFoundException ex) {  
103             Arquivo.escreve_log("Erro ao ler dados dos alunos salvos no arquivo "  
104                 ARQUIVO_BINARIO_ALUNOS);  
105         }  
106     } catch (IOException e) {  
107         Arquivo.escreve_log("Erro ao criar o arquivo " + ARQUIVO_BINARIO_ALUNOS);  
108     }  
109     return alunos;  
110 }  
111
```

# Relembrando...

```
46 public static void main(String[] args) throws IOException {  
47     Aluno[] alunos = new Aluno[TOTAL_ALUNOS];  
48     SetorEnsino ensino = new SetorEnsino(DIRETOR_ENSINO, COORDENADOR_ENSINO);  
49     int opcao = 4;
```



```
41 public static void main(String[] args) {  
42     Aluno[] alunos = Arquivo.obtem_alunos();  
43     SetorEnsino ensino = Arquivo.obtem_ensino();  
44     int opcao = 4;
```

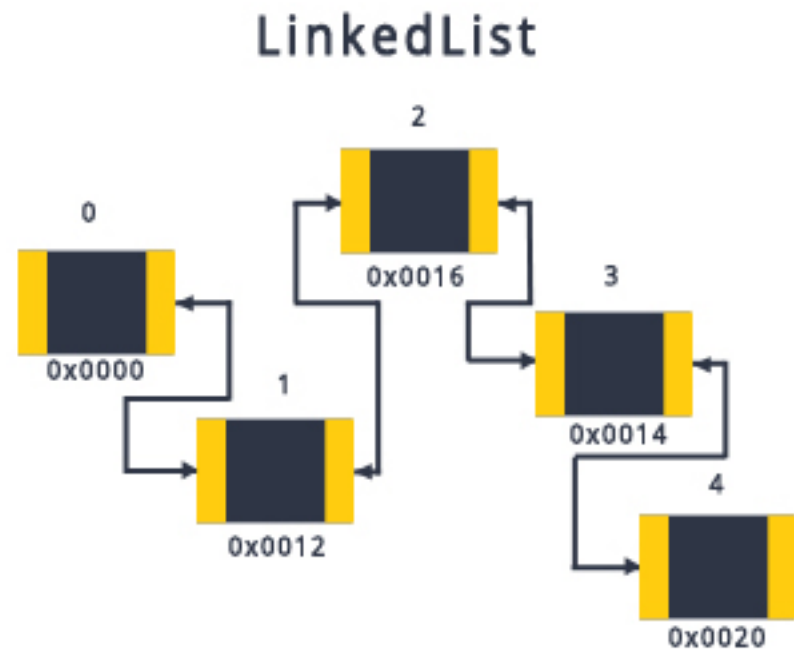
# Relembrando...

```
110     }  
111     } while (opcao != OP_SAIR);  
112     Arquivo.salva_alunos(alunos);  
113     Arquivo.salva_ensino(ensino);  
114 }
```

# Estruturas de Dados

## Listas

RAFAEL VIEIRA COELHO



# Visão geral de Estrutura de Dados

Estrutura de Dados	Vantagens	Desvantagens
Vetor	Inserção rápida, acesso muito rápido se o índice for conhecido	Pesquisa lenta, remoção lenta, tamanho fixo.
Fila	Fornece acesso do tipo primeiro a entrar ultimo a sair.	Acesso lento para outros itens.
Pilha	Fornece acesso do tipo ultimo a entrar, primeiro a sair.	Acesso lento para outros itens.
Lista ligada	Inserção rápida, remoção rápida	Pesquisa lenta.
Grafo	Modela situações do mundo real.	Alguns algoritmos são lentos e complexos.
Árvore binária	Pesquisa, inserção e remoção rápidas.	O algoritmo de remoção é complexo.

# Vetor (Array) x ArrayList

- A primeira estrutura que tivemos contato foram os arrays (vetores);
- No entanto, eles são limitados devido a capacidade fixa (não pode ser redimensionado);
- Além disso, remover elementos geram a necessidade de deslocar parte do conteúdo do array;
- A ordenação de um array não é uma tarefa simples.
- Podemos contornar estes problemas usando um ArrayList:

```
ArrayList<String> name = new ArrayList<String>();
```

# 1) ArrayList

- As listas são estruturas de dados de armazenamento sequencial assim como os arrays;
- Mas diferentemente dos arrays, as listas não possuem capacidade fixa o que facilita bastante o trabalho.

```
ArrayList<Integer> v = new ArrayList();
```

```
int vi = v.get(2); //vi = v[2]
```

```
v.set(10, 2); // v[2] = 10
```

```
v.add(20); //??
```

```
int t = v.size(); // t = v.length
```

primitive	wrapper class
int	Integer
boolean	Boolean
double	Double
char	Character



# Métodos Úteis de um ArrayList

add( <b>value</b> )	myList.add("hello")
add( <b>index, value</b> )	myList.add(0, "hello")
clear()	myList.clear()
indexOf( <b>value</b> )	myList.indexOf("hello")
get( <b>index</b> )	myList.get(0)
remove( <b>index</b> )	myList.remove(0)
set( <b>index, value</b> )	myList.set(0, "goodbye")
size()	myList.size()
toString()	myList.toString()

# Exemplo de Uso do Add e Remove

size = 2

0	1	2	3	4	5	6	7
4	7	1	0	0	0	0	...

add(4) ;

add(7) ;

add(1) ;

-> current size = 2

-> add this at index 2

-> update size to 3

remove(2) ;

-> update size to 2

# Percorrendo um ArrayList

```
ArrayList<Integer> myList = new ArrayList<Integer>();  
int sum = 0;  
for (int i = 0; i < myList.size(); i++) {  
    sum += myList.get(i);  
}
```

- Podemos usar o For-Each também quando não queremos alterar o ArrayList:

```
ArrayList<Integer> v = new ArrayList();
```

```
for (Integer n : v) {  
    System.out.println(n);  
}
```

# Interface List

- List é a interface Java que define os métodos que uma lista deve implementar.
- As principais implementações da interface List são:
  1. ArrayList;
  2. LinkedList;
  3. Vector.
- Cada implementação possui suas características sendo apropriadas para contextos diferentes.

# Criação de Listas (new) e Adição(add)

```
16 public class Criacao {  
17     public static void main(String[] args) {  
18         ArrayList<Aluno> lista1 = new ArrayList();  
19         lista1.add(new Aluno("Sandra", 1212));  
20         lista1.add(new Aluno("Jorge", 2222));  
21  
22         for (Aluno aluno : lista1) {  
23             System.out.println(aluno.toString());  
24         }  
25  
26         ArrayList<Professor> lista2 = new ArrayList();  
27         lista2.add(new Professor("Prof. Sandro", 32121));  
28         lista2.add(new Professor("Prof. Maria", 21214));  
29  
30         for (Professor professor : lista2) {  
31             System.out.println(professor.toString());  
32         }  
33     }
```

# Tamanho da Lista: size()

```
15 public class Tamanho {  
16     public static void main(String[] args) {  
17         ArrayList<Professor> lista2 = new ArrayList();  
18  
19         lista2.add(new Professor("Prof. Sandro", 32121));  
20         lista2.add(new Professor("Prof. Maria", 21214));  
21  
22         System.out.println("Número de Professores: " + lista2.size());  
23     }  
24 }  
25
```

# Remoção de Todos Elementos: clear()

```
15 public class Limpar {
16
17     public static void main(String[] args) {
18         ArrayList<Aluno> lista1 = new ArrayList();
19
20         lista1.add(new Aluno("Sandra", 1212));
21         lista1.add(new Aluno("Jorge", 2222));
22         mostra_lista(lista1);
23
24         lista1.clear();
25         mostra_lista(lista1);
26     }
27
28     private static void mostra_lista(ArrayList<Aluno> lista1) {
29         System.out.println("\n ===== Lista ===== ");
30         for (Aluno aluno : lista1) {
31             System.out.println(aluno.toString());
32         }
33         System.out.println(" =====");
34     }
35
36 }
```

# Verificando a Existência de um Elemento na Lista: contains(Object)

```
15 public class Pesquisa {  
16     public static void main(String[] args) {  
17         ArrayList<Aluno> lista1 = new ArrayList();  
18         Aluno jorge = new Aluno("Jorge", 2222);  
19  
20         lista1.add(new Aluno("Sandra", 1212));  
21         lista1.add(jorge);  
22  
23         if (lista1.contains(jorge)) {  
24             System.out.println("O aluno jorge está na lista");  
25         } else {  
26             System.out.println("O aluno jorge não está na lista");  
27         }  
28     }  
29 }  
30 }
```



# Remoção de Elementos: remove(Object) e remove(int)

```
15 public class Remocao {
16
17     public static void main(String[] args) {
18         ArrayList<Aluno> lista1 = new ArrayList();
19         Aluno sandra = new Aluno("Sandra", 1212);
20
21         lista1.add(sandra);
22         lista1.add(new Aluno("Jorge", 2222));
23         mostra_lista(lista1);
24         lista1.remove(sandra);
25         mostra_lista(lista1);
26         lista1.remove(0);
27         mostra_lista(lista1);
28     }
29
30     private static void mostra_lista(ArrayList<Aluno> lista1) {
31         System.out.println("\n ===== Lista ===== ");
32         for (Aluno aluno : lista1) {
33             System.out.println(aluno.toString());
34         }
35         System.out.println(" =====");
36     }
37 }
```

# Pesquisa de Elementos na Lista: get(int)

```
15 public class Obtencao {  
16     public static void main(String[] args) {  
17         ArrayList<Aluno> lista1 = new ArrayList();  
18  
19         lista1.add(new Aluno("Sandra", 1212));  
20         lista1.add(new Aluno("Jorge", 2222));  
21  
22         Aluno sandra = lista1.get(0);  
23         Aluno jorge = lista1.get(1);  
24  
25         System.out.println(sandra);  
26         System.out.println(jorge);  
27     }  
28 }
```

# Pesquisa de Elementos na Lista: indexOf(Object)

```
15 public class Indice {  
16     public static void main(String[] args) {  
17         ArrayList<Aluno> lista1 = new ArrayList();  
18         Aluno sandra = new Aluno("Sandra", 1212);  
19  
20         lista1.add(new Aluno("Jorge", 2222));  
21         lista1.add(sandra);  
22  
23         int indice_sandra = lista1.indexOf(sandra);  
24  
25         System.out.println("Posição: " + indice_sandra);  
26     }  
27 }  
28
```

# Ordenação de uma Lista

```
ArrayList<Integer> lista = new ArrayList();
```

```
lista.add(8);
```

```
lista.add(3);
```

```
lista.add(5);
```

```
lista.add(2);
```

```
Collections.sort(lista);
```

```
// A lista contém os elementos na ordem {2, 3, 5, 8} depois da chamada  
da função sort
```

# Ordenando uma Lista de Alunos

```
public static void main(String[] args) {
```

```
    ArrayList<Aluno> lista = new ArrayList();
```

```
    lista.add(new Aluno("Sandra", 1));
```

```
    lista.add(new Aluno("Jorge", 2));
```

```
    lista.add(new Aluno("Sandro", 3));
```

```
    lista.add(new Aluno("Maria", 4));
```

```
    lista.add(new Aluno("Sandro", 5));
```

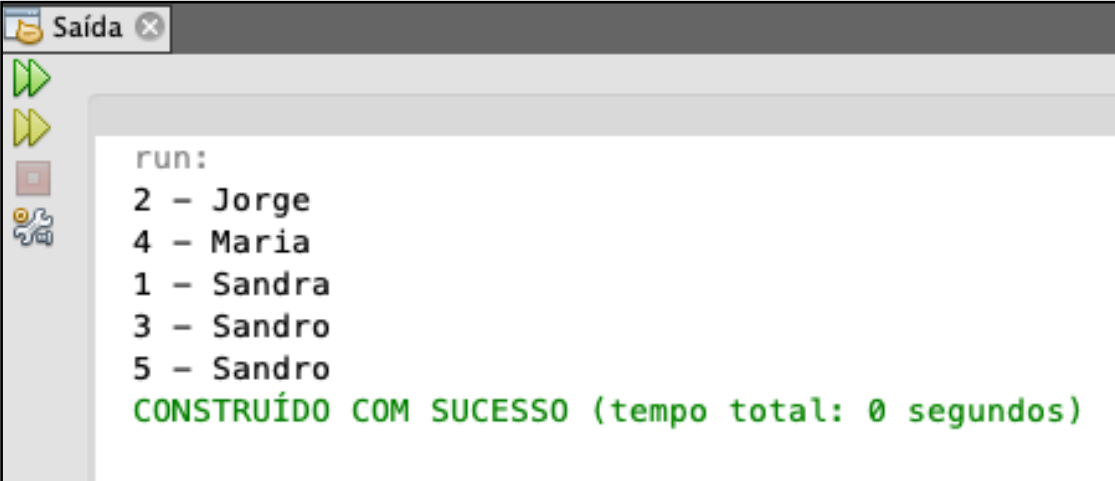
```
    Collections.sort(lista);
```

```
    for (Aluno aluno : lista) {
```

```
        System.out.println(aluno.getMatricula() + " - " + aluno.getNome());
```

```
    }
```

```
}
```



The screenshot shows a Java IDE's output window titled 'Saída'. It displays the output of a program that sorts a list of students by their matriculation number. The output is as follows:

```
run:
2 - Jorge
4 - Maria
1 - Sandra
3 - Sandro
5 - Sandro
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

# Enquanto isto, na classe Aluno

```
public class Aluno implements Serializable, Comparable<Aluno> {  
    ...  
    @Override  
    public int compareTo(Aluno outro) {  
        Long mat1 = this.matricula;  
        Long mat2 = outro.matricula;  
  
        return this.nome.compareTo(outro.nome) + mat1.compareTo(mat2);  
    }  
}
```

O método compareTo deve retornar 0 se forem iguais.  
Ou o quão diferentes são.

# Tarefas

- 1) Faça com que todas as classes implementem a interface Comparable.
- 2) Modifique toda ocorrência de arrays para ArrayList (onde são declarados e usados).