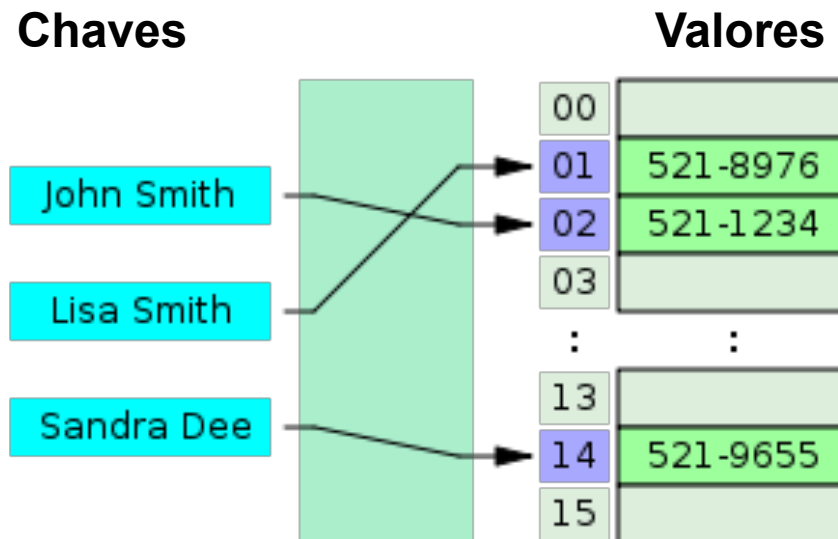


Estruturas de Dados

3) Mapas

RAFAEL VIEIRA COELHO



Relembrando...

1) Faça com que todas as classes implementem a interface Comparable.

```
14 public class Curso implements Comparable {
15
16     public static final int MAX_DISCIPLINAS = 40;
17     private String nome;
18     private String ppc;
19     private Disciplina disciplinas[];
20
21     @Override
22     public int compareTo(Object outro) {
23         return this.nome.compareTo(((Curso) outro).nome);
24     }
```

Relembrando...

1) Faça com que todas as classes implementem a interface Comparable.

```
9      public class Professor implements Comparable {
10
11          private String nome;
12          private String areas[];
13          private long siape;
14
15          @Override
16          public int compareTo(Object outro) {
17              Long s1 = this.siape;
18              Long s2 = ((Professor) outro).siape;
19
20              return this.nome.compareTo(((Professor) outro).nome)
21                  + s1.compareTo(s2);
22          }
}
```

Relembrando...


1) Faça com que todas as classes implementem a interface Comparable.

```
9 public class Disciplina implements Comparable {
10
11     public static final int MAX_ALUNOS = 30;
12
13     private Aluno alunos[];
14     private Professor professor;
15     private String nome;
16     private int ano;
17     private float notas[];
18
19     @Override
20     public int compareTo(Object outro) {
21         return this.nome.compareTo(((Disciplina) outro).nome)
22             + this.professor.compareTo(((Disciplina) outro).professor);
23     }
```

Relembrando...

2) Modifique toda ocorrência de arrays para ArrayList (onde são declarados e usados).

```
19 private Disciplina disciplinas[];  
20  
21 public Curso() {  
22  
23 }  
24  
25 public boolean novaDisciplina(String nome, int ano, Professor professor)  
26     for (int i = 0; i < disciplinas.length; i++) {  
27         if (disciplinas[i] != null) {  
28             disciplinas[i] = new Disciplina(professor, nome, ano);  
29             return true;  
30         }  
31     }  
32     return false;  
33 }
```



Relembrando...


2) Modifique toda ocorrência de arrays para ArrayList (onde são declarados e usados).

```
18     private String nome;
19     private String ppc;
20     private ArrayList<Disciplina> disciplinas;
21
22     @Override
23     public int compareTo(Object outro) {
24         return this.nome.compareTo(((Curso) outro).nome);
25     }
26
27     public Curso() {
28
29     }
30
31     public boolean novaDisciplina(String nome, int ano, Professor profes
32         return disciplinas.add(new Disciplina(professor, nome, ano));
33     }
```

Relembrando...

2) Modifique toda ocorrência de arrays para ArrayList (onde são declarados e usados).

```
35  public boolean removerDisciplina(String nome) {  
36      for (int i = 0; i < disciplinas.length; i++) {  
37          if (disciplinas[i] != null  
38              && disciplinas[i].getNome().equals(nome)) {  
39              disciplinas[i] = null;  
40              return true;  
41          }  
42      }  
43      return false;  
44  }  
45  
46  public Disciplina[] getDisciplinas() {  
47      return disciplinas;  
48  }  
49
```



Relembrando...

2) Modifique toda ocorrência de arrays para ArrayList (onde são declarados e usados).


```
35  public boolean removerDisciplina(String nome) {  
36      for (Iterator<Disciplina> iterator = disciplinas.iterator();  
37          iterator.hasNext();) {  
38          Disciplina d = iterator.next();  
39  
40          if (d != null && d.getNome().equals(nome)) {  
41              iterator.remove();  
42              return true;  
43          }  
44      }  
45      return false;  
46  }
```

```
64  public ArrayList<Disciplina> getDisciplinas() {  
65      return disciplinas;  
66  }
```


Relembrando...

2) Modifique toda ocorrência de arrays para ArrayList (onde são declarados e usados).

```
50  [-] public void setDisciplinas(Disciplina[] disciplinas) {  
51      this.disciplinas = disciplinas;  
52  }  
53  
54  [-] public Curso(String nome, String ppc, Disciplina[] disciplinas) {  
55      this.nome = nome;  
56      this.ppc = ppc;  
57      this.disciplinas = disciplinas;  
58  }  
59  
60  [-] public Curso(String nome, String ppc) {  
61      this.nome = nome;  
62      this.ppc = ppc;  
63      this.disciplinas = new Disciplina[MAX_DISCIPLINAS];  
64  }
```

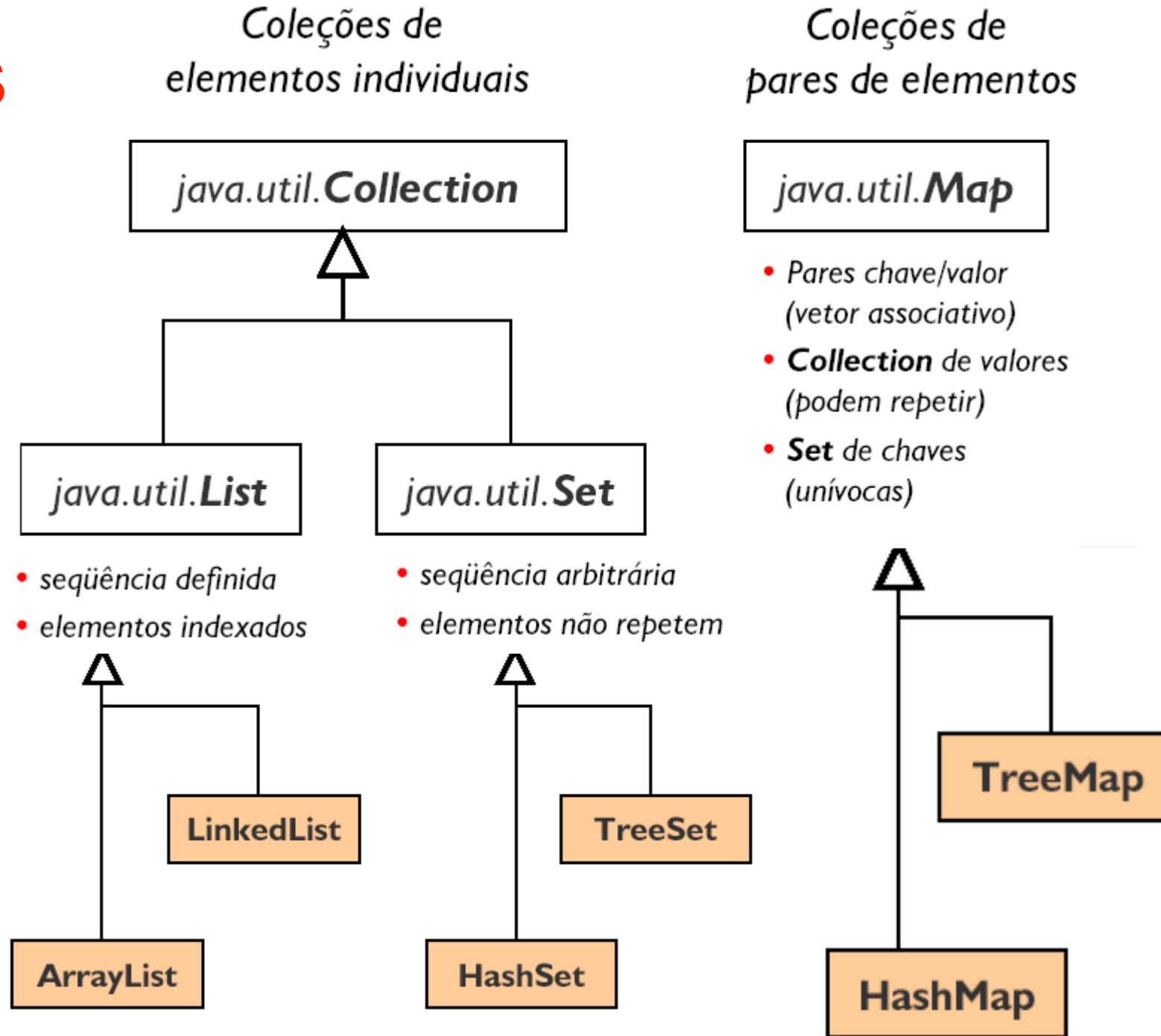


Relembrando...

2) Modifique toda ocorrência de arrays para ArrayList (onde são declarados e usados).

```
68  public void setDisciplinas(ArrayList<Disciplina> disciplinas) {
69      this.disciplinas = disciplinas;
70  }
71
72  public Curso(String nome, String ppc,
73      ArrayList<Disciplina> disciplinas) {
74      this.nome = nome;
75      this.ppc = ppc;
76      this.disciplinas = disciplinas;
77  }
78
79  public Curso(String nome, String ppc) {
80      this.nome = nome;
81      this.ppc = ppc;
82      this.disciplinas = new ArrayList();
83  }
```

Tipos de Coleções em Java



Iterando em Listas

```
List<String> names = new ArrayList();  
names.add("Clementine");  
names.add("Duran");  
names.add("Mike");
```

//Java 1.2

```
Iterator<String> listIterator = names.iterator();  
while (listIterator.hasNext()) {  
    System.out.println(listIterator.next());  
}
```

Iterando em Listas

//Java 5

```
for (int i = 0; i < names.size(); i++) {  
    System.out.println(names.get(i));  
}
```

//Java 5

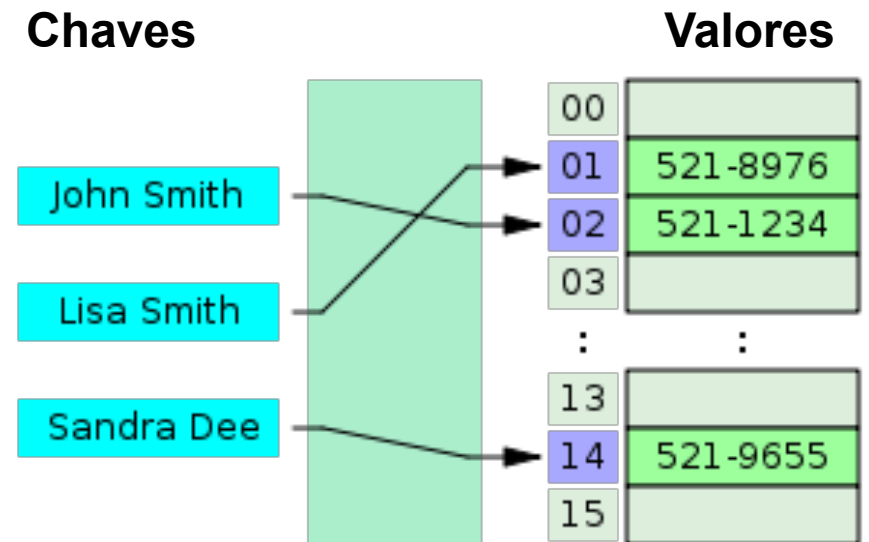
```
for (String name : names) {  
    System.out.println(name);  
}
```

//Java 8

```
names.forEach(System.out::println);
```

HashMap: Definição

- O Map tem uma combinação de chaves/valores.
- Funciona da seguinte maneira: uma chave está associada a um valor.
- As chaves, claro, são únicas.
- Os valores podem ser duplicados.



HashMap: Principais Métodos

- `put(Object key, Object value)` – acrescenta um objeto na lista, associando-o a uma chave.
- `get(Object key)` - retorna um objeto através de sua chave.
- `keySet()` - retorna um Set com as chaves
- `values()` - retorna uma Collection com os valores.

Iterando em Mapas

```
Map<Integer, String> names2 = new HashMap();  
names2.put(1, "Clementine");  
names2.put(2, "Duran");  
names2.put(3, "Mike");
```

//Java 5

```
for (Map.Entry<Integer, String> entry : names2.entrySet()) {  
    System.out.println(entry.getKey());  
    System.out.println(entry.getValue());  
}
```


Iterando em Mapas

//Java 5

```
for (Integer key : names2.keySet()) {  
    System.out.println(key);  
}  
  
for (String value : names2.values()) {  
    System.out.println(value);  
}
```

Iterando em Mapas

//Java 5

```
Iterator entries = names2.entrySet().iterator();  
while (entries.hasNext()) {  
    Map.Entry entry = (Map.Entry) entries.next();  
    System.out.println(entry.getKey());  
    System.out.println(entry.getValue());  
}
```

//Java 8

```
names2.forEach((key, value) -> System.out.println("Key: " + key + " Value: " +  
value));
```

Criação

```
16 public class Criacao {
17
18     public static void main(String[] args) {
19         HashMap<Aluno, Float> mapa1 = new HashMap();
20
21         mapa1.put(new Aluno("Sandra", 1212), 10.0f);
22         mapa1.put(new Aluno("Jorge", 2222), 5.6f);
23
24         for (Map.Entry<Aluno, Float> entry : mapa1.entrySet()) {
25             Aluno key = entry.getKey();
26             Float value = entry.getValue();
27
28             System.out.println("A nota do(a) aluno(a) "
29                               + key.getNome()
30                               + " é "
31                               + value);
32         }
33     }
34 }
35
```

Tamanho

```
15 public class Tamanho {  
16  
17     public static void main(String[] args) {  
18         HashMap<Aluno, Float> mapa1 = new HashMap();  
19         Aluno sandra = new Aluno("Sandra", 1212);  
20         Aluno jorge = new Aluno("Jorge", 2222);  
21  
22         mapa1.put(sandra, 10.0f);  
23         mapa1.put(jorge, 5.6f);  
24         System.out.println("Número de Alunos: " + mapa1.size());  
25     }  
26 }
```

Obtenção

```
15 public class Obtencao {
16
17     public static void main(String[] args) {
18         HashMap<Aluno, Float> mapa1 = new HashMap();
19         Aluno sandra = new Aluno("Sandra", 1212);
20         Aluno jorge = new Aluno("Jorge", 2222);
21
22         mapa1.put(sandra, 10.0f);
23         mapa1.put(jorge, 5.6f);
24
25         float nota = mapa1.get(jorge);
26
27         System.out.println("Nota: " + nota);
28     }
29 }
```

Alteração

```
16 public class Alteracao {
17
18     public static void main(String[] args) {
19         HashMap<Aluno, Float> mapa1 = new HashMap();
20         Aluno sandra = new Aluno("Sandra", 1212);
21         Aluno jorge = new Aluno("Jorge", 2222);
22
23         mapa1.put(sandra, 10.0f);
24         mapa1.put(jorge, 5.6f);
25         mostra_mapa(mapa1);
26         mapa1.replace(jorge, 8.9f);
27         mapa1.replace(sandra, 6.7f);
28         mostra_mapa(mapa1);
29     }
30 }
```

Remoção

```
19 public static void main(String[] args) {  
20     HashMap<Aluno, Float> mapa1 = new HashMap();  
21     Aluno sandra = new Aluno("Sandra", 1212);  
22     Aluno jorge = new Aluno("Jorge", 2222);  
23  
24     mapa1.put(sandra, 10.0f);  
25     mapa1.put(jorge, 5.6f);  
26     mostra_mapa(mapa1);  
27     mapa1.remove(jorge);  
28     mostra_mapa(mapa1);  
29     for (Iterator<Aluno> iterator = mapa1.keySet().iterator();  
30         iterator.hasNext();) {  
31         Aluno aluno = iterator.next();  
32  
33         if (aluno.equals(sandra)) {  
34             iterator.remove();  
35         }  
36     }
```

OBSERVAÇÃO ALEATÓRIA

- Obtendo informações do sistemas operacional

```
public class ExemploDeProperties {  
    public static void main(String[] args) {  
        Properties p = System.getProperties();  
        System.out.println("Todas as propriedades do sistema operacional:");  
        System.out.println(p);  
        System.out.println("Imprimindo propriedades isoladas:");  
        System.out.println("Nome do SO: " + p.get("os.name"));  
        System.out.println("Nome do Runtime: " + p.get("java.runtime.name"));  
        System.out.println("País do usuário: " + p.get("user.country"));  
    }  
}
```


OBSERVAÇÃO ALEATÓRIA

Todas as propriedades do sistema operacional:

```
{java.runtime.name=Java(TM) SE Runtime Environment, sun.boot.library.path=/home/
coelho/jdk1.8.0_121/jre/lib/amd64, java.vm.version=25.121-b13, java.vm.vendor=Oracle
Corporation, java.vendor.url=http://java.oracle.com/, path.separator=:,
java.vm.name=Java HotSpot(TM) 64-Bit Server VM, file.encoding.pkg=sun.io,
user.country=BR, sun.java.launcher=SUN_STANDARD, sun.os.patch.level=unknown,
java.vm.specification.name=Java Virtual Machine Specification, user.dir=/home/coelho/
Dropbox/IFRS 2017/Disciplina_Programação 2 (INTEGRADO)/AULAS/2o Trimestre/
Exemplo_Colecoes, java.runtime.version=1.8.0_121-b13,...
```

Imprimindo propriedades isoladas:

Nome do SO: Linux

Nome do Runtime: Java(TM) SE Runtime Environment

País do usuário: BR

Tarefas

- 1) Transforme os dois ArrayLists de alunos e notas em um mapa (HashMap).