Criar um projeto -> Aula14.
Criar no pacote entity as classes Aluno e Endereco. Na classe Aluno implementar a interface Comparable, criar os atributos (idAluno, nome, disciplina, nota1,nota2 e media, sendo transient), relacionar com a classe Endereco. Depois criar os construtores vazio e cheio sem os atributos de media e relacionamento de Endereço. Colocar o toString e getters e setters. Inicializar o atributo média com 0 (zero), pois será calculada pelo sistema.

```
package entity;

import java.util.ArrayList;
import java.util.List;

public class Aluno implements Comparable<Aluno> {

	private Integer idAluno;
	private String nome;
	private String disciplina;
	private Double nota1;
	private Double nota2;
	private transient Double media = 0.;

	private Endereco endereco;

	public Aluno() {

	}

	public Aluno(Integer idAluno, String nome, String
disciplina, Double nota1, Double nota2) {
	super();
	this.idAluno = idAluno;
	this.nome = nome;
	this.disciplina = disciplina;
	this.nota1 = nota1;
	this.nota2 = nota2;
    }


	@Override
	public String toString() {
		return "Aluno [idAluno=" + idAluno + ", nome=" + nome +
", disciplina=" + disciplina + ", nota1=" + nota1 + ", nota2=" +
nota2 + ", media=" + media + ", endereco=" + endereco + "]";
	}
```

```
public Integer getIdAluno() {
	return idAluno;
}
public void setIdAluno(Integer idAluno) {
	this.idAluno = idAluno;
}
public String getNome() {
	return nome;
}
public void setNome(String nome) {
	this.nome = nome;
}
public String getDisciplina() {
	return disciplina;
}
public void setDisciplina(String disciplina) {
	this.disciplina = disciplina;
}
public Double getNota1() {
	return nota1;
}
public void setNota1(Double nota1) {
	this.nota1 = nota1;
}
public Double getNota2() {
	return nota2;
}
public void setNota2(Double nota2) {
	this.nota2 = nota2;
}
public Double getMedia() {
	return media;
}
public void setMedia(Double media) {
	this.media = media;
}
public Endereco getEndereco() {
	return endereco;
}
public void setEndereco(Endereco endereco) {
	this.endereco = endereco;
}
```

```
@Override
public boolean equals(Object obj) {
	Aluno a = (Aluno) obj;
	return this.idAluno.equals(a.getIdAluno());
}


@Override
public int compareTo(Aluno a) {
	return  this.idAluno.compareTo(a.getIdAluno());
}


@Override
public int hashCode() {
	return   String.valueOf(idAluno).length();
}
```

No método main, criar 2 alunos e 2 endereços. Chamando a classe Aluno, criando o objeto e dando espaço de memória. Fazer o mesmo para a classe Endereco. Relacionar os objetos de Aluno setando os objetos de Endereco.

```
public static void main(String[] args) {
	Aluno al1 = new Aluno(10,"jose","java",7.,8.);
	Aluno al2 = new Aluno(11,"carlos","oracle",9.,8.);

	Endereco end1= new Endereco(100,"vila
Emil","Mesquita");
	Endereco end2= new Endereco(200,"lapa","RJ");
		al1.setEndereco(end1);
		al2.setEndereco(end2);
```

Chamar a interface List, passar Aluno como parâmetro e criar o objeto de lista com o espaço de memória para trabalhar. Chamar o objeto lista e o método add com o parâmetro do objeto aluno. Pedir para imprimir o aluno da lista que está na posição 0 (zero). O mesmo com o endereco, irá imprimir o endereco constante na lista na posição 0 (zero).

```
List<Aluno> lista =  new ArrayList<Aluno>();
		lista.add(al1);
		lista.add(al2);
 System.out.println("Aluno: " + lista.get(0));
```

```
            System.out.println("Resposta: " +
lista.get(0).getEndereco());
      }
}
```

No console...

```
Aluno: Aluno [idAluno=10, nome=jose, disciplina=java, nota1=7.0,
nota2=8.0, media=0.0, endereco=Endereco [idEndereco=100,
bairro=lapa Emil, cidade=Mesquita]]

Resposta: Endereco [idEndereco=100, bairro=lapa Emil,
cidade=Mesquita]
```

Na classe Endereco criar os atributos (idEndereco,bairro, cidade). Na classe Endereco relacionar com a classe Aluno, criar o construtor vazio, o cheio sem o atributo de relacionamento do aluno, toString (sem os dados de aluno) e os getters e setters.

```
package entity;

public class Endereco {
      private    Integer idEndereco;
      private    String bairro;
      private    String cidade;

      private    Aluno aluno;

      public Endereco() {

      }


      public Endereco(Integer idEndereco, String bairro, String
cidade) {
            super();
            this.idEndereco = idEndereco;
            this.bairro = bairro;
            this.cidade = cidade;
      }
```

```
	@Override
	public String toString() {
		return "Endereco [idEndereco=" + idEndereco + ",
bairro=" + bairro + ", cidade=" + cidade + "]";
	}


	public Integer getIdEndereco() {
		return idEndereco;
	}
	public void setIdEndereco(Integer idEndereco) {
		this.idEndereco = idEndereco;
	}
	public String getBairro() {
		return bairro;
	}
	public void setBairro(String bairro) {
		this.bairro = bairro;
	}
	public String getCidade() {
		return cidade;
	}
	public void setCidade(String cidade) {
		this.cidade = cidade;
	}
	public Aluno getAluno() {
		return aluno;
	}
	public void setAluno(Aluno aluno) {
		this.aluno = aluno;
	}
}
```

Criar um pacote chamado mapa e nele criar uma classe chamada Mapa. Criar um método de teste, main.

```
package mapa;

import java.util.Map;
import java.util.TreeMap;
import entity.Aluno;
```

```
import entity.Endereco;

public class Mapa {

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

Chamar a interface Mapa, tipar os objetos a serem criado como Integer e String e depois criar o objeto mapa. Dar espaço através do TreeMap, passando os mesmos tipos. Depois chamar o objeto e o método put, para inserir os dados no mapa, que são: código e nome.

```
          Map<Integer, String> mapa = new TreeMap<Integer,
String>();
             mapa.put(10, "luis");
             mapa.put(8, "andre");
             mapa.put(12, "lidia");
```

Criamos um loop de for onde tipamos o objeto chave e chamamos o mapa e usamos o keySet. Pedimos para imprimir essa chave que virá ordenada. Depois fazemos o mesmo para imprimirmos a lista dos nomes que também virá ordenada.

```
             System.out.println("Chaves: ");
             for (Integer chave : mapa.keySet()){
                  System.out.println(chave);
             }

             System.out.println("Nomes: ");
             for (String texto : mapa.values()){
                  System.out.println(texto);
             }
```

Nessa impressão escrevemos um tetxto (“Boolena”) e contacatenamos com o objeto mapa e o método containsKey onde tipamos para Integer e verificamos se existe o numero 10 de chave. Esse tipo de método retorna um booleano (true ou false). Se 10 existir a resposat será verdadeira (true).

```
          System.out.println("Booleana :" + mapa.containsKey(new
Integer(10)));
```

Nesse novo teste criamos uma variável de String onde será inciada com o espaço entre as aspas e uma outra variável Integer com valor "10". No loop de for tipamos Integer para a variável "chave" e chamamos o objeto "mapa" com o método keySet e damos a condição se a variável "chaveproc" for igual a "chave", a "chaveproc" recebera o valor de "chave". A variavel "procuravalor" receberá do "mapa" na posição da "chave" o valor de String correspondente. Onde imprimiremos o valor da chave e seu conteúdo,

```
String procuravalor = "";
Integer chaveproc  = 10;

for (Integer chave : mapa.keySet()){
     if (chaveproc.equals(chave)){
          chaveproc = chave;
          procuravalor = mapa.get(chaveproc);
        }
 }
  System.out.println("Chave=" + chaveproc + ", Conteudo
:" + procuravalor);
```

Nesse novo teste pela inteface Map passamos as classe Aluno e Endereco como parâmetros para o objeto "mapa2" e damos espaço com o TreeMap. Adicionamos em "mapa2", através do put os dados de Aluno e Endereco. Pedimos para imprimir esse objeto

```
  Map  <Aluno, Endereco> mapa2 = new
TreeMap<Aluno,Endereco>();
mapa2.put(new Aluno(10,"jose","java",10.,6.),
            new Endereco(1,"mesquita","mesquita"));
mapa2.put(new Aluno(12,"carlos","ORACLE",9.,9.),
            new Endereco(1,"tijuca","rj"));
System.out.println(mapa2);
```

Novo teste com a classe Aluno e objeto "busca" dar espaço de memória. Setar o código do aluno e criar uma condição de que se se no mapa2 contiver o código passado em busca cai no loop do for, onde passo para o objeto de Aluno a chave keySey de mapa2.

Na condição if, se o resultado da busca for igual ao objeto, passo que busca recebra a, o objeto, e busca setará o endereço. Senão, se a busca for nula, vazia, imprime a mensagem.

Se passar pela condição, imprime o nome do aluno e depois imprime o endereço.

```
            Aluno busca = new Aluno();
                  busca.setIdAluno(10);
             if (mapa2.containsKey(busca)){
              for(Aluno a : mapa2.keySet()){
                    if (busca.equals(a)){
                        busca = a;
                        busca.setEndereco(mapa2.get(busca));
                    }
                }
             }else{
              busca=null;
              System.out.println("Nao encontrado");
             }
          System.out.println("Nome do Aluno :" + busca.getNome());
          System.out.println("Dados do Endereco:" +
busca.getEndereco());
      }
}
```

No console...

```
Chaves:
8
10
12

Nomes:
andre
luis
lidia

Booleana :true

Chave=10, Conteudo :Luis

{Aluno [idAluno=10, nome=jose, disciplina=java, nota1=10.0,
nota2=6.0, media=0.0, endereco=null]=Endereco [idEndereco=1,
bairro=mesquita, cidade=mesquita], Aluno [idAluno=12,
nome=carlos, disciplina=ORACLE, nota1=9.0, nota2=9.0, media=0.0,
```

```
endereco=null]=Endereco [idEndereco=1, bairro=tijuca,
cidade=rj]}

Nome do Aluno :jose
Dados do Endereco:Endereco [idEndereco=1, bairro=mesquita,
cidade=mesquita]
```