Criar um projeto -> Aula12.
Criar no pacote entity a classe Usuario com os atributos(idUsuario, nomeUsuario, login, senha). Implementar a interface Comparable. Criar construtor vazio, o construtor cheio, o toString sem a impressão da senha e os getters e setters.

```
package entity;

import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;
import java.util.List;


public class Usuario implements Comparable<Usuario>{

	private Integer idUsuario;
	private String  nomeUsuario;
	private String  login;
	private String  senha;

	public Usuario() {

	}

	public Usuario(Integer idUsuario, String nomeUsuario, String
login, String senha) {
		super();
		this.idUsuario = idUsuario;
		this.nomeUsuario = nomeUsuario;
		this.login = login;
		this.senha = senha;
	}
```

Na criação do toString, não incluir o atributo senha pois senão será impresso a senha.

```
	@Override
	public String toString() {
		return "Usuario [idUsuario=" + idUsuario + ",
nomeUsuario="  + nomeUsuario + ", login=" + login +"]";
	}
```

```
public Integer getIdUsuario() {
      return idUsuario;
}
public void setIdUsuario(Integer idUsuario) {
      this.idUsuario = idUsuario;
}
public String getNomeUsuario() {
      return nomeUsuario;
}
public void setNomeUsuario(String nomeUsuario) {
      this.nomeUsuario = nomeUsuario;
}
public String getLogin() {
      return login;
}
public void setLogin(String login) {
      this.login = login;
}
public String getSenha() {
      return senha;
}
public void setSenha(String senha) {
      this.senha = senha;
}
```

Criamos o método equals com a classe Object e damos as seguintes condições: se o objeto passado for igual a esse, retornará verdadeiro. Senão, se o objeto for nulo, retornará falso. Se não, se o objeto for diferente de Usuario, retorna falso. Ou senão, passamos a classe Usuario com seu objeto e fazemos um casting transformando o Usuario no objeto obj, tendo como retorno a comparação entre os códigos dos usuários.

```
public boolean equals(Object obj){
 if (obj == this){
      return true;
 }else if (obj == null){
      return false;
 }else if (!(obj instanceof Usuario)){
      return false;
 }else{
      Usuario u  = (Usuario) obj;
      return this.idUsuario.equals(u.getIdUsuario());
 }
```

```
}
```

No método hashCode, iremos ordenar pela quantidade de dígitos contidos no código dos usuários. Passamos a classe String e criamos uma variável quantidade,

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

No método compareTo, será comparado os códigos, o código já existente com o código passado.

```
 @Override
 public int compareTo(Usuario u) {
      return this.idUsuario.compareTo(u.getIdUsuario());
 }
```

Criamos o metodo main para testar. Nele chamamos a classe Usuario e passamos seu objeto. Depois damos espaço de memória e passamos os dados para a criação dos usuários.

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

Usuario u2 = new Usuario(100,"jose","jose@gmail.com","23432");
Usuario u3 = new Usuario(99,"carlos","carlos@gmail.com","5555");
Usuario u4 = new Usuario(20,"jo","jo@gmail.com","123");
Usuario u5 = new Usuario(30,"joao","joao@gmail.com","546");
Usuario u6 = new Usuario(300,"lu","lu@gmail.com","546");
Usuario u7 = new Usuario(1,"pedro","pedro@gmail.com","546");
```

Através da interface List passamos o Usuario como parâmetro, criamos o objeto lst e damos espaço para lista trabalhar. Pelo objeto lst adicionamos os objetos dos usuários. Chamamos a classe Collection e o método sort e

passamos o objeto da lista. Em seguida pedimos para imprimir esse objeto com o resultado da lista já ordenado pelo código.

```
List<Usuario> lst = new ArrayList<Usuario>();
  lst.add(u2);
  lst.add(u3);
  lst.add(u4);
  lst.add(u5);
Collections.sort(lst);
System.out.println(lst);
```

Já na interface do HashSet passamos a classe usuário e um objeto lst2. Damos espaço para trabalhar. Chamamos o objeto e adicionamos os usuários. Depois imprimimos esse objeto ordenado pela quantdade de caracteres.

```
HashSet<Usuario> lst2 = new HashSet<>();
     lst2.add(u2);
     lst2.add(u3);
     lst2.add(u4);
     lst2.add(u5);
     lst2.add(u6);
     lst2.add(u7);
System.out.println(lst2);
    }
}
```

No console...

```
[Usuario [idUsuario=20, nomeUsuario=jo, login=jo@gmail.com],
Usuario [idUsuario=30, nomeUsuario=joao, login=joao@gmail.com],
Usuario [idUsuario=99, nomeUsuario=carlos,
login=carlos@gmail.com], Usuario [idUsuario=100,
nomeUsuario=jose, login=jose@gmail.com]]

[Usuario [idUsuario=1, nomeUsuario=pedro,
login=pedro@gmail.com], Usuario [idUsuario=30, nomeUsuario=joao,
login=joao@gmail.com], Usuario [idUsuario=20, nomeUsuario=jo,
login=jo@gmail.com], Usuario [idUsuario=99, nomeUsuario=carlos,
login=carlos@gmail.com], Usuario [idUsuario=300, nomeUsuario=lu,
```

```
login=lu@gmail.com], Usuario [idUsuario=100, nomeUsuario=jose,
login=jose@gmail.com]]
```

Escola de Nerds
www.cotiinformatica.com.br