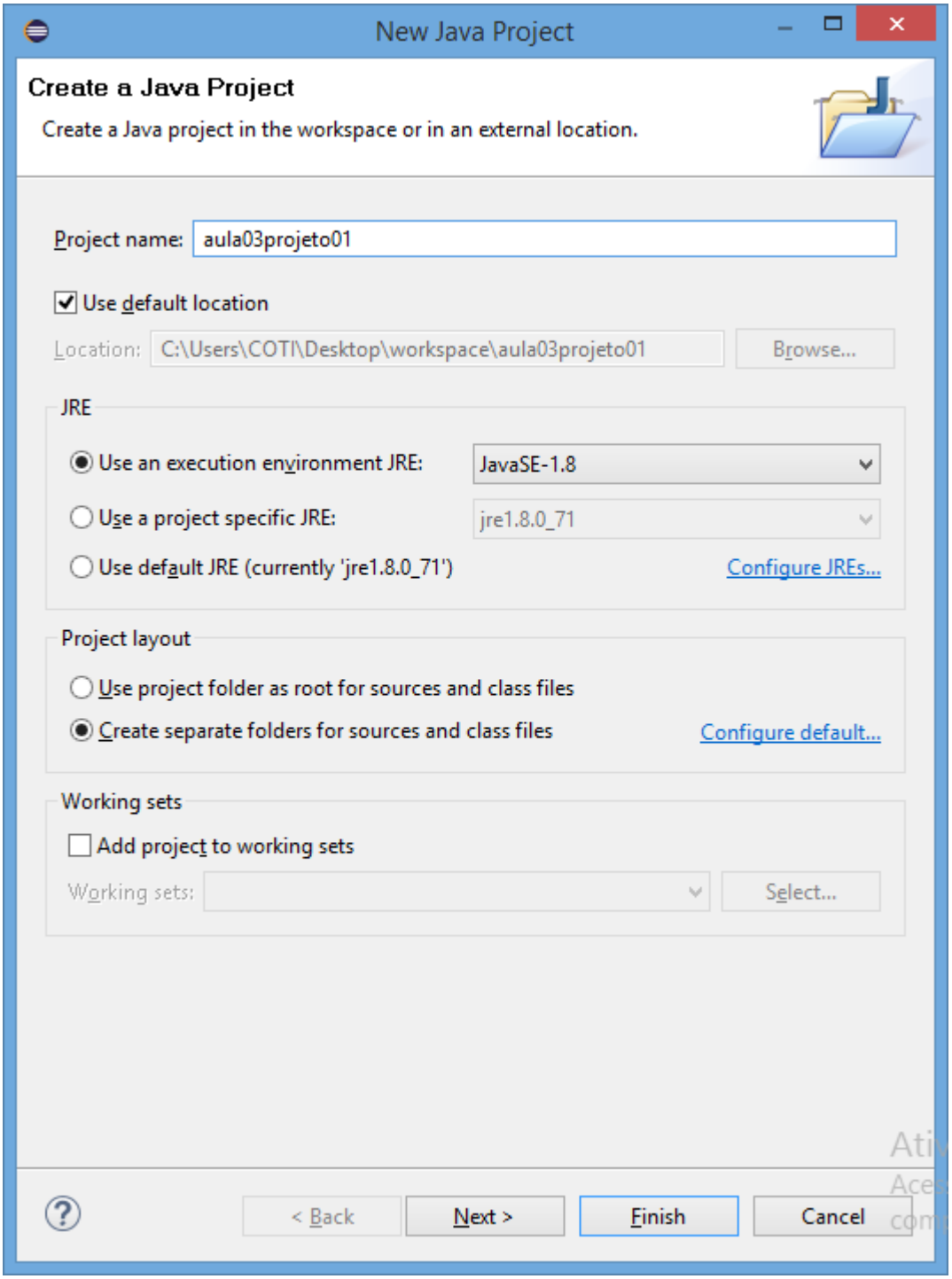
Criando um novo projeto:

- File > New > Project



Criando Enums

```
package entities.types;

public enum EstadoCivil {

	Solteiro,
	Casado,
	Divorciado,
	Viuvo
}
```

```
package entities.types;

public enum Sexo {

	Masculino,
	Feminino
}
```

Criando uma entidade Pessoa:

/entities/Pessoa.java

```
package entities;

import entities.types.EstadoCivil;
import entities.types.Sexo;

//classe javabean esta implementando a interface Comparable (sort, ordenação)
//quando implementamos Comparable, devemos programar o método compareTo
public class Pessoa implements Comparable<Pessoa>{

	private Integer idPessoa;
	private String nome;
	private Sexo sexo;
	private EstadoCivil estadoCivil;

	public Pessoa() {
		// TODO Auto-generated constructor stub
	}

	public Pessoa(Integer idPessoa, String nome, Sexo sexo, EstadoCivil estadoCivil) {
		this.idPessoa = idPessoa;
		this.nome = nome;
		this.sexo = sexo;
		this.estadoCivil = estadoCivil;
	}
```

```
public Integer getIdPessoa() {
	return idPessoa;
}

public void setIdPessoa(Integer idPessoa) {
	this.idPessoa = idPessoa;
}

public String getNome() {
	return nome;
}

public void setNome(String nome) {
	this.nome = nome;
}

public Sexo getSexo() {
	return sexo;
}

public void setSexo(Sexo sexo) {
	this.sexo = sexo;
}

public EstadoCivil getEstadoCivil() {
	return estadoCivil;
}

public void setEstadoCivil(EstadoCivil estadoCivil) {
	this.estadoCivil = estadoCivil;
}

//Sobresrita dos metodos da classe object...
//equals -> define regra de comparação para objetos da classe
@Override //sobrescrita de metodo
public boolean equals(Object obj) {

	if(obj instanceof Pessoa){

		Pessoa p = (Pessoa) obj;

		if(p.getIdPessoa() != null){
			return p.getIdPessoa().equals(idPessoa);
		}
	}
	return false;
}
```

```
	//hashCode -> define regra para ordenação e classificação
	//geralmente a regra do hashcode é a mesma do equals
	@Override //sobrescrita de metodo
	public int hashCode() {
		return idPessoa.hashCode(); //ordenação pelo id..
	}

	//toString -> retorna uma string com o valor do objeto
	@Override //sobrescrita de metodo
	public String toString() {
		return idPessoa + ", " + nome + ", " + sexo + ", " + estadoCivil;
	}

	//método da interface Comparable
	//utilizado para definir regras de ordenação em collections
	@Override
	public int compareTo(Pessoa p) {

		//regra de comparação pelo id..
		//if(p.getIdPessoa() != null){
			//ordem decrescente...
			//return p.getIdPessoa().compareTo(idPessoa);
		//}

		//ordem crescente...
		return nome.compareTo(p.getNome());
	}
}
```

Comparable<T>

Existe uma interface especial em Java que é responsável por fazer comparações de objetos, que é a classe *Comparable*. Ela usada como padrão de comparação.

Essa comparação é feita através do método *compareTo(Object o*), que recebe um objeto (se lembre que todos os objeto são derivados da classe *Object*, logo, todo e qualquer objeto que é possível criar em Java é derivado de *Object*).

Collections

Coleções de dados

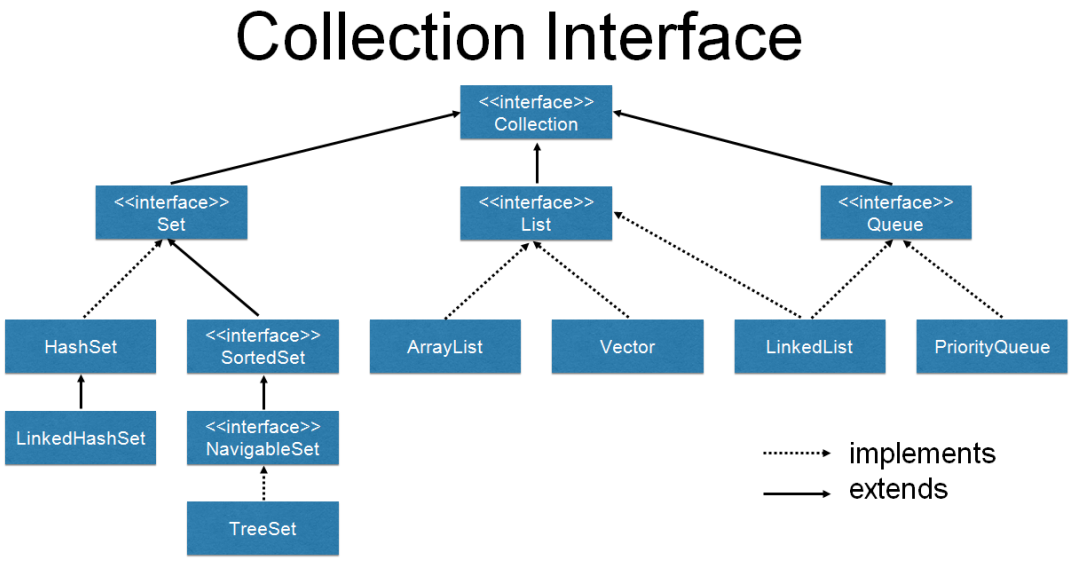
Uma Collection É um objeto onde podemos agrupar vários elementos. No dia-a-dia nos deparamos com várias situações onde as coleções estão presentes: uma ***fila*** de banco, uma ***lista*** de compras, uma ***pilha*** de livros, um ***conjunto*** de elementos, etc.

- ***Collection:*** O *framework* não possui implementação direta desta *inferface*, porém, ela está no topo da hierarquia definindo operações que são comuns a todas as coleções;

- ***Set:*** Está diretamente relacionada com a idéia de conjuntos. Assim como um conjunto, as classes que implementam esta interface não podem conter elementos repetidos. Usaremos implementações de *SortedSet* para situações onde desejarmos ordenar os elementos;

- ***List:*** Também chamada de seqüência. É uma coleção ordenada, que ao contrário da inferface *Set,* pode conter valores duplicados. Além disso, temos controle total sobre a posição onde se encontra cada elemento de nossa coleção, podendo acessar cada um deles pelo índice.

- ***Queue:*** Normalmente utilizamos esta interface quando queremos uma coleção do tipo FIFO (First-In-First-Out), também conhecida como fila.

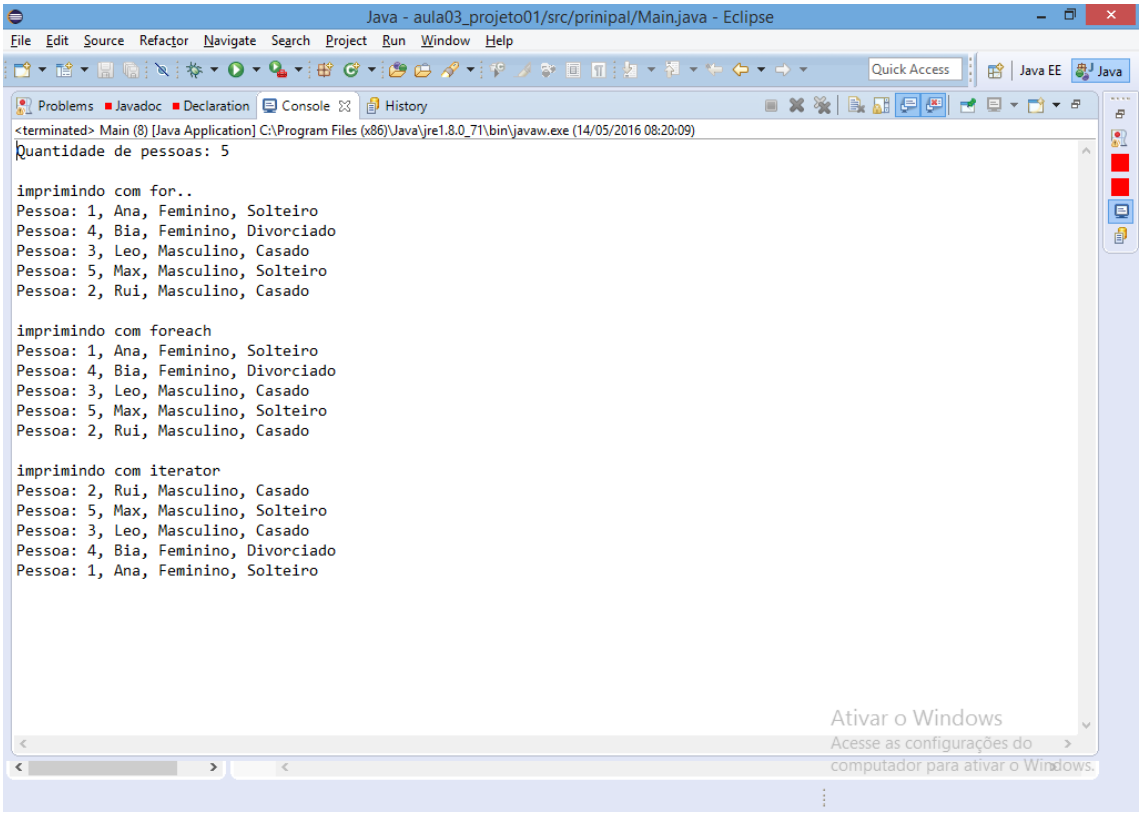


Manipulando listas:

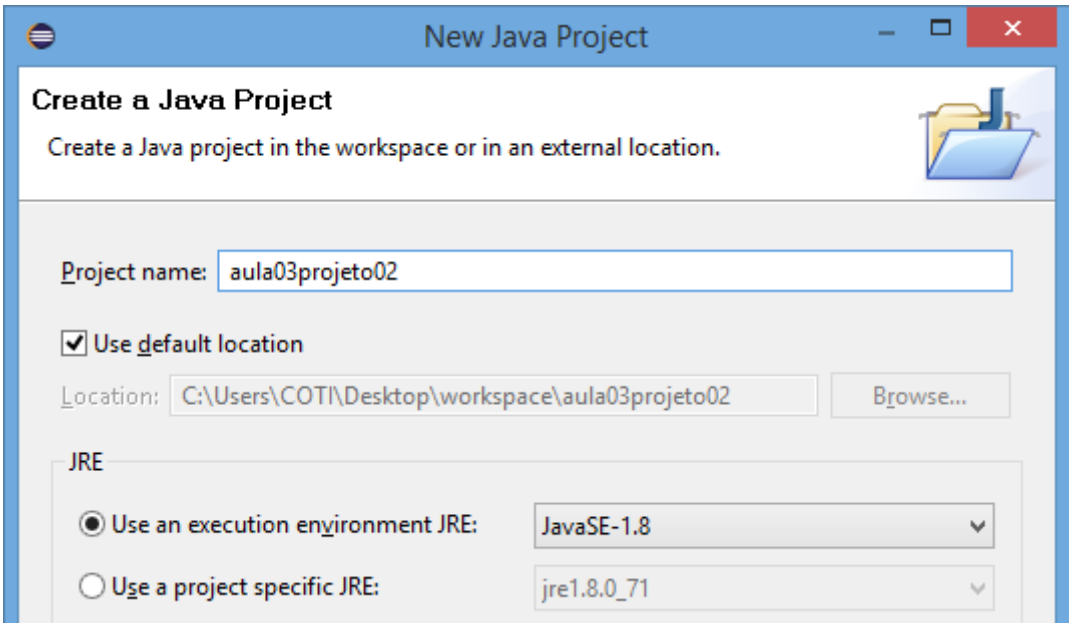
```
package prinipal;

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

import entities.Pessoa;
import entities.types.EstadoCivil;
import entities.types.Sexo;

public class Main {

	public static void main(String[] args) {

		//criando objetos..
		Pessoa p1 = new Pessoa(1, "Ana", Sexo.Feminino, EstadoCivil.Solteiro);
		Pessoa p2 = new Pessoa(2, "Rui", Sexo.Masculino, EstadoCivil.Casado);
		Pessoa p3 = new Pessoa(3, "Leo", Sexo.Masculino, EstadoCivil.Casado);
		Pessoa p4 = new Pessoa(4, "Bia", Sexo.Feminino, EstadoCivil.Divorciado);
		Pessoa p5 = new Pessoa(5, "Max", Sexo.Masculino, EstadoCivil.Solteiro);

		//declarando uma lista de pessoas..
		List<Pessoa> lista = new ArrayList<Pessoa>();

		//adicionando objetos na lista..
		lista.add(p1); //adicionando..
		lista.add(p2); //adicionando..
		lista.add(p3); //adicionando..
		lista.add(p4); //adicionando..
		lista.add(p5); //adicionando..

		//imprimindo a quantidade de pessoas na lista..
		System.out.println("Quantidade de pessoas: " + lista.size());


		//recurso de ordenação para listas..
		//em collections, temos um método de ordenação chamado de sort
		//para que a lista possa ser ordenada pelo metodo sort, é necessario
		//que a classe que define o tipo da lista implemente uma
```

```
        //interface chamada de Comparable
        Collections.sort(lista);

        //percorrendo a lista..
        System.out.println("\nimprimindo com for..");
        for(int i = 0; i < lista.size(); i++){
            //recuperar uma pessoa dentro da lista..
            Pessoa p = lista.get(i);
            //imprimindo..
            System.out.println("Pessoa: " + p); //toString()
        }

        System.out.println("\nimprimindo com foreach");
        for(Pessoa p : lista){
            //imprimindo..
            System.out.println("Pessoa: " + p); //toString()
        }

        //podemos transformar a lista em um iterator (cursor)...
        //e imprimi-la de uma forma diferente (mais antigo no java)

        //colocar a lista em ordem inversa..
        Collections.reverse(lista);

        //convertendo a lista em um interator..
        Iterator<Pessoa> iterator = lista.iterator();

        //um iterator nao é percorrido com for ou foreach, mas sim com while..
        System.out.println("\nimprimindo com iterator");
        while(iterator.hasNext()){
            //obter o elemento do iterator..
            Pessoa p = iterator.next();
            //imprimindo..
            System.out.println("Pessoa: " + p); //toString()
        }
    }
}
```

Executando:
Saída do programa...



Novo projeto:



Criando uma entidade Produto:

```
package entities;

import entities.types.Status;

public class Produto implements Comparable<Produto>{

	private Integer idProduto;
	private String nome;
	private Double preco;
	private Integer quantidade;
	private Status status;

	public Produto() {
		// TODO Auto-generated constructor stub
	}

	public Produto(Integer idProduto, String nome,
			Double preco, Integer quantidade, Status status) {
		this.idProduto = idProduto;
		this.nome = nome;
		this.preco = preco;
		this.quantidade = quantidade;
		this.status = status;
	}

	public Integer getIdProduto() {
		return idProduto;
	}

	public void setIdProduto(Integer idProduto) {
		this.idProduto = idProduto;
	}

	public String getNome() {
		return nome;
	}

	public void setNome(String nome) {
		this.nome = nome;
	}

	public Double getPreco() {
		return preco;
	}

	public void setPreco(Double preco) {
		this.preco = preco;
	}
```

```
	public Integer getQuantidade() {
		return quantidade;
	}

	public void setQuantidade(Integer quantidade) {
		this.quantidade = quantidade;
	}

	public Status getStatus() {
		return status;
	}

	public void setStatus(Status status) {
		this.status = status;
	}

	@Override
	public boolean equals(Object obj) {

		if(obj instanceof Produto){

			Produto p = (Produto) obj;

			if(p.getIdProduto() != null){
				return p.getIdProduto().equals(idProduto);
			}
		}
		return false;
	}

	@Override
	public int hashCode() {
		return idProduto.hashCode();
	}

	@Override
	public String toString() {
		return idProduto + ", " + nome + ", " + preco
				+ ", " + quantidade + ", " + status;
	}

	@Override
	public int compareTo(Produto p) {
		return nome.compareTo(p.getNome());
	}

}
```

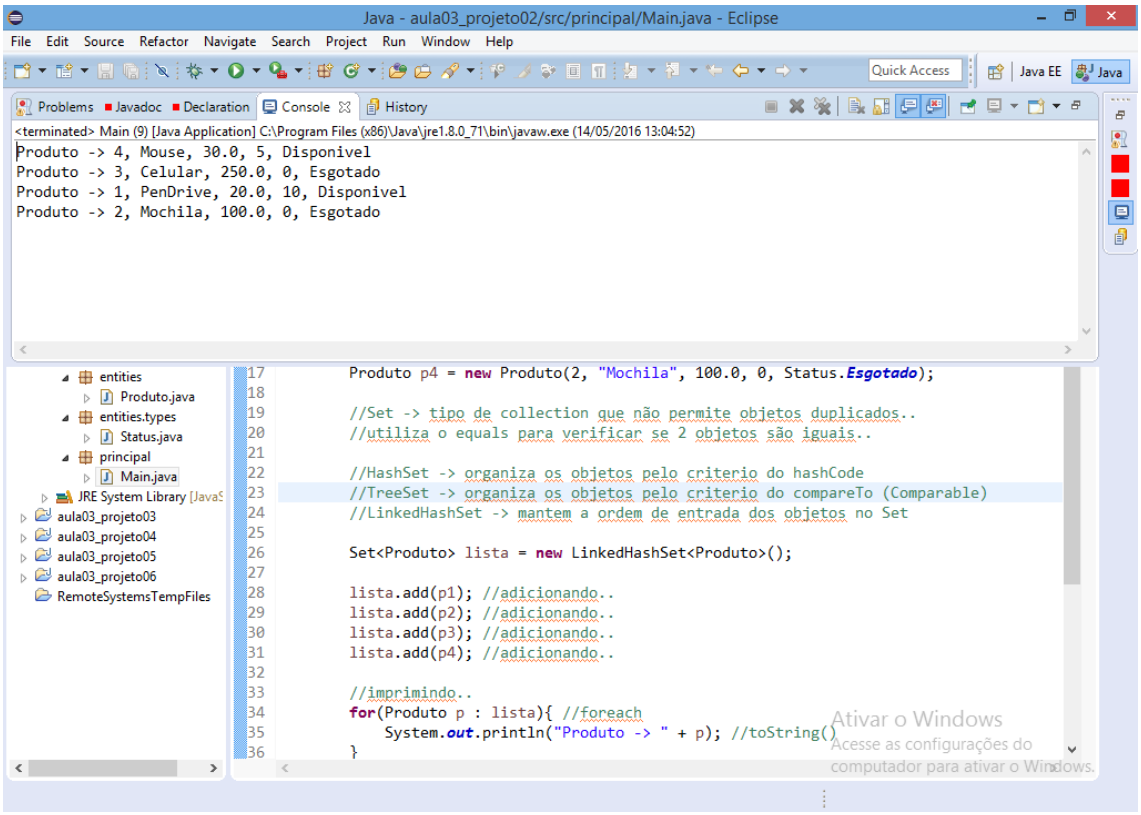
```
package entities.types;

public enum Status {

	Disponivel,
	Esgotado

}
```

Utilizando a interface Set<T>

```
package principal;

import java.util.LinkedHashSet;
import java.util.Set;

import entities.Produto;
import entities.types.Status;

public class Main {

	public static void main(String[] args) {

		//criando objetos...
		Produto p1 = new Produto(4, "Mouse", 30.0, 5, Status.Disponivel);
		Produto p2 = new Produto(3, "Celular", 250.0, 0, Status.Esgotado);
		Produto p3 = new Produto(1, "PenDrive", 20.0, 10, Status.Disponivel);
		Produto p4 = new Produto(2, "Mochila", 100.0, 0, Status.Esgotado);

		//Set -> tipo de collection que não permite objetos duplicados..
		//utiliza o equals para verificar se 2 objetos são iguais..

		//HashSet -> organiza os objetos pelo criterio do hashCode
		//TreeSet -> organiza os objetos pelo criterio do compareTo (Comparable)
		//LinkedHashSet -> mantem a ordem de entrada dos objetos no Set

		Set<Produto> lista = new LinkedHashSet<Produto>();

		lista.add(p1); //adicionando..
		lista.add(p2); //adicionando..
		lista.add(p3); //adicionando..
		lista.add(p4); //adicionando..
```

```
                //imprimindo..
                for(Produto p : lista){ //foreach
                        System.out.println("Produto -> " + p); //toString()
                }
        }
}
```

Executando:

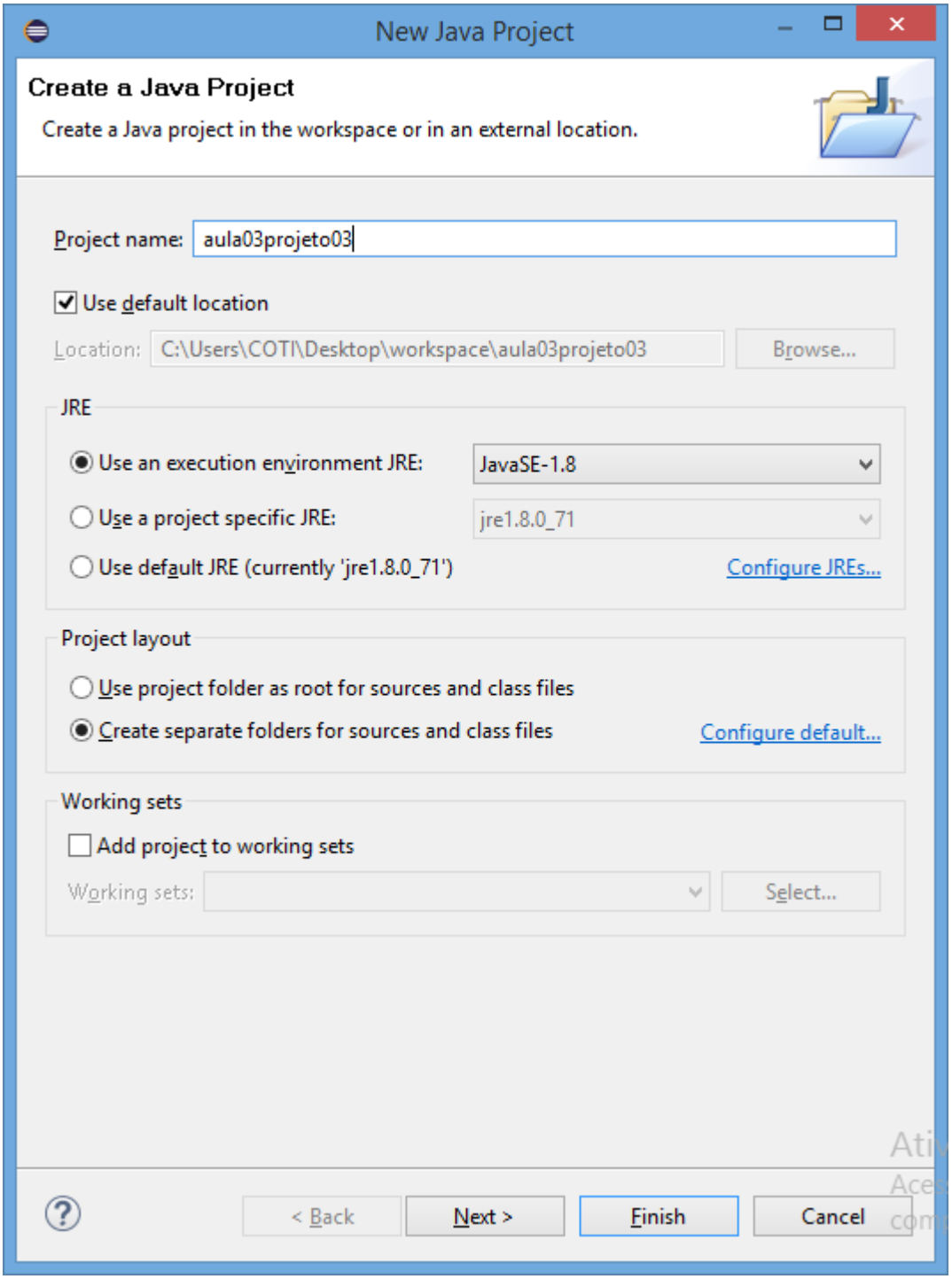


É importante notar que TreeSet, HashSet e LinkedHashSet implementam a interface “Set”, ou seja, temos os mesmos métodos para as 3 estruturas, o que difere cada uma é a forma com que é implementado o algoritmo, por exemplo, na HashSet já falamos que ela usa HashTable em sua implementação, que por sinal é muito rápido mas não garante a ordenação dos seus elementos

O TreeSet implementa um algoritmo conhecido por red-black tree ou árvore rubro-negra. Sua principal característica é que ele é o único Set que implementa a interface SortedSet em vez de Set diretamente, mas de qualquer forma SortedSet implementa Set, assim continuamos tendo os mesmo métodos no TreeSet.

Novo projeto:

- File / New / Project



Criando uma entidade Produto:

```
package entities;

public class Produto implements Comparable<Produto>{

	private Integer idProduto;
	private String nome;
	private Double preco;

	public Produto() {
		// TODO Auto-generated constructor stub
	}

	public Produto(Integer idProduto, String nome, Double preco) {
		this.idProduto = idProduto;
		this.nome = nome;
		this.preco = preco;
	}

	public Integer getIdProduto() {
		return idProduto;
	}

	public void setIdProduto(Integer idProduto) {
		this.idProduto = idProduto;
	}

	public String getNome() {
		return nome;
	}

	public void setNome(String nome) {
		this.nome = nome;
	}

	public Double getPreco() {
		return preco;
	}

	public void setPreco(Double preco) {
		this.preco = preco;
	}

	@Override
	public String toString() {
		return "Produto [idProduto=" + idProduto
			+ ", nome=" + nome + ", preco=" + preco + "]";
	}
```

```
	@Override
	public int hashCode() {
		return idProduto.hashCode();
	}

	@Override
	public boolean equals(Object obj) {

		//assertiva -> condição ? entao : senao
		return obj instanceof Produto
				? ((Produto)obj).getIdProduto() != null
				? ((Produto)obj).getIdProduto()
				.equals(idProduto)
				: false : false;
	}

	@Override
	public int compareTo(Produto p) {
		return idProduto.compareTo(p.getIdProduto());
	}

}
```

Criando uma classe de controle para manipular uma lista de Produtos:

```
package control;

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

import entities.Produto;

public class ControleProduto {

	//atributo -> lista de produtos..
	private List<Produto> listagemProdutos; //null

	//construtor..
	public ControleProduto() {
		//inicializar o atributo da classe..
		listagemProdutos = new ArrayList<Produto>();
	}
```

```
//método para adicionar um produto na lista..
public void adicionarProduto(Produto p){
	listagemProdutos.add(p);
}

//método para buscar um produto dentro da lista pelo id..
public Produto buscarPorId(Integer idProduto){
	//varrendo a lista.
	for(Produto p : listagemProdutos){
		//verificando se o produto da lista tem o mesmo id do parametro..
		if(p.getIdProduto().equals(idProduto)){
			return p; //retornando o produto..
		}
	}
	return null; //vazio..
}

//método para retornar produtos pelo nome (contendo)..
public List<Produto> buscarPorNome(String nome){

	//criando uma lista vazia..
	List<Produto> lista = new ArrayList<Produto>();

	//varrer a listagem de produtos da classe..
	for(Produto p : listagemProdutos){
		//verificar se o produto contem o nome passado no metodo..
		if(p.getNome().toLowerCase().contains(nome.toLowerCase())){
			lista.add(p); //adicionar na lista de resposta..
		}
	}

	//retornar a lista..
	return lista;
}


//método para limpar a lista..
public void limparListagemProdutos(){
	listagemProdutos.clear();
}
```

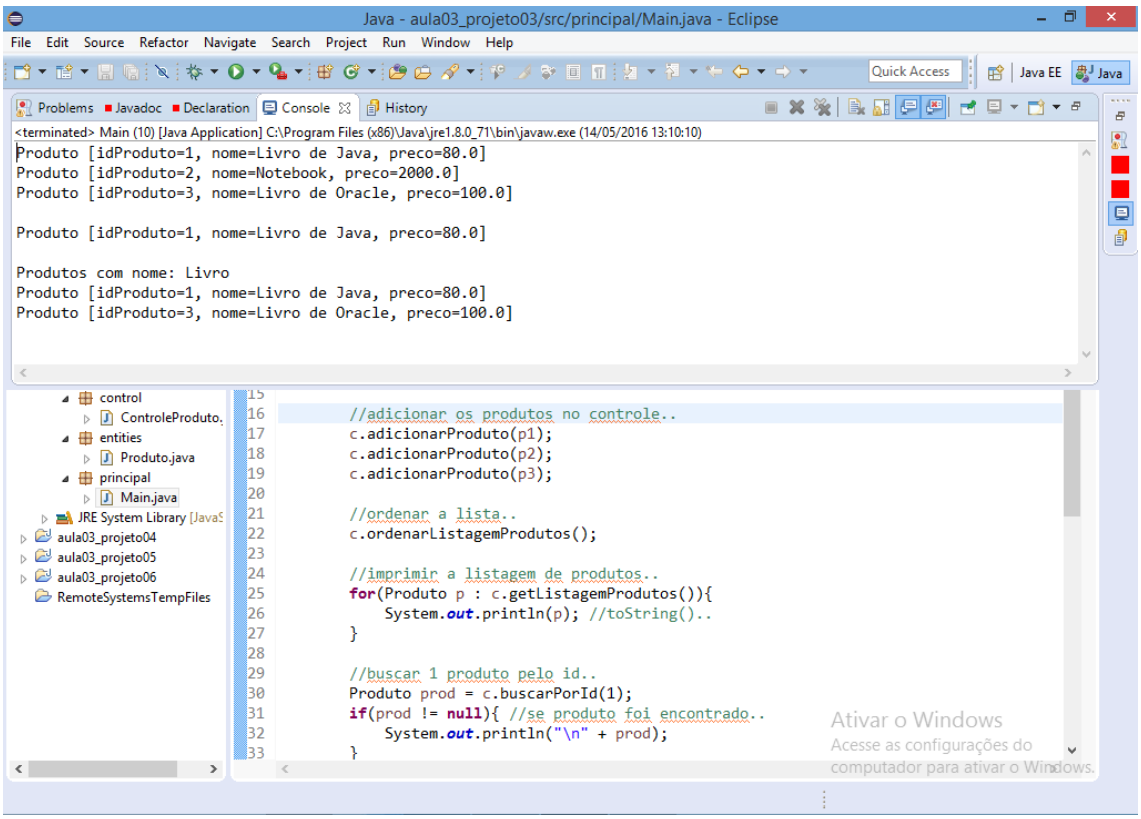
```
        //método para ordenar a lista..
        public void ordenarListagemProdutos(){
                Collections.sort(listagemProdutos);
        }


        //método para retornar a listagem de produtos..
        public List<Produto> getListagemProdutos() {
                return listagemProdutos;
        }
}
```

Executando:
Método Main

```
package principal;

import control.ControleProduto;
import entities.Produto;

public class Main {

        public static void main(String[] args) {

                ControleProduto c = new ControleProduto(); //classe de controle..

                Produto p1 = new Produto(1, "Livro de Java", 80.0);
                Produto p2 = new Produto(3, "Livro de Oracle", 100.0);
                Produto p3 = new Produto(2, "Notebook", 2000.0);

                //adicionar os produtos no controle..
                c.adicionarProduto(p1);
                c.adicionarProduto(p2);
                c.adicionarProduto(p3);

                //ordenar a lista..
                c.ordenarListagemProdutos();

                //imprimir a listagem de produtos..
                for(Produto p : c.getListagemProdutos()){
                        System.out.println(p); //toString()..
                }
```

```
                //buscar 1 produto pelo id..
                Produto prod = c.buscarPorId(1);
                if(prod != null){ //se produto foi encontrado..
                        System.out.println("\n" + prod);
                }
                else{
                        System.out.println("\nProduto não encontrado.");
                }

                System.out.println("\nProdutos com nome: Livro");
                //buscar produtos pelo nome..
                for(Produto p : c.buscarPorNome("Livro")){
                        System.out.println(p); //imprimindo..
                }
        }
}
```

Saida do programa:



Novo projeto:

```
package entities;

public class Funcionario implements Comparable<Funcionario>{

	private Integer idFuncionario;
	private String nome;
	private Double salario;

	public Funcionario() {
		// TODO Auto-generated constructor stub
	}

	public Funcionario(Integer idFuncionario,
			String nome, Double salario) {
		super();
		this.idFuncionario = idFuncionario;
		this.nome = nome;
		this.salario = salario;
	}

	public Integer getIdFuncionario() {
		return idFuncionario;
	}

	public void setIdFuncionario(Integer idFuncionario) {
		this.idFuncionario = idFuncionario;
	}

	public String getNome() {
		return nome;
	}

	public void setNome(String nome) {
		this.nome = nome;
	}

	public Double getSalario() {
		return salario;
	}

	public void setSalario(Double salario) {
		this.salario = salario;
	}

	@Override
	public int hashCode() {
		return idFuncionario.hashCode();
	}
```

```
	@Override
	public boolean equals(Object obj) {

		if(obj instanceof Funcionario){
			Funcionario f = (Funcionario) obj;
			if(f.getIdFuncionario() != null){
				return f.getIdFuncionario()
					.equals(idFuncionario);
			}
		}

		return false;
	}

	@Override
	public String toString() {
		return "Funcionario [idFuncionario=" + idFuncionario
			+ ", nome=" + nome + ", salario=" + salario + "]";
	}

	@Override
	public int compareTo(Funcionario f) {
		return idFuncionario.compareTo(f.getIdFuncionario());
	}
}
```

Criando uma interface para definir um contrato de operações com Funcionario:

```
package contracts;

import java.util.List;

import entities.Funcionario;

//nivel abstrato (abstração)
public interface IControleFuncionario {

	//adicionar 1 funcionario na lista
	void adicionarFuncionario(Funcionario f);

	//limpar o conteudo da lista
	void limparListagemFuncionarios();

	//ordenar a lista
	void ordenarListagemFuncionarios();
```

```
        //buscar 1 Funcionario pelo id dentro da lista
        Funcionario obterPorId(Integer idFuncionario);

        //buscar funcionarios pelo nome dentro da lista
        List<Funcionario> obterPorNome(String nome);

        //buscar funcionarios pelo salario dentro da lista
        List<Funcionario> obterPorSalario(Double salarioIni, Double salarioFim);

        //retornar todos os funcionarios da lista
        List<Funcionario> getListagemFuncionarios();
}
```

Implementando a interface:

```
package control;

import java.util.ArrayList;
import java.util.Collections;
import java.util.LinkedHashSet;
import java.util.List;
import java.util.Set;

import contracts.IControleFuncionario;
import entities.Funcionario;

public class ControleFuncionario implements IControleFuncionario {

        //atributo..
        private Set<Funcionario> lista;

        //construtor..
        public ControleFuncionario() {
                //LinkedHashSet -> mantem a ordem de entrada dos objetos..
                lista = new LinkedHashSet<Funcionario>();
        }

        @Override
        public void adicionarFuncionario(Funcionario f) {
                lista.add(f); //adicionando..
        }

        @Override
        public void limparListagemFuncionarios() {
                lista.clear();
        }
```

```
@Override
public void ordenarListagemFuncionarios() {

	List<Funcionario> auxiliar = new ArrayList<Funcionario>(lista);
	Collections.sort(auxiliar); //ordenação..

	lista = new LinkedHashSet<Funcionario>(auxiliar);
}

@Override
public Funcionario obterPorId(Integer idFuncionario) {

	for(Funcionario f : lista){
		if(f.getIdFuncionario().equals(idFuncionario)){
			return f; //retornar o funcionario..
		}
	}

	return null; //vazio..
}

@Override
public List<Funcionario> obterPorNome(String nome) {

	List<Funcionario> resultado = new ArrayList<Funcionario>();

	for(Funcionario f : lista){
		if(f.getNome().toUpperCase().contains(nome.toUpperCase())){
			resultado.add(f);
		}
	}

	return resultado;
}

@Override
public List<Funcionario> obterPorSalario(Double salarioIni, Double salarioFim) {

	List<Funcionario> resultado = new ArrayList<Funcionario>();

	for(Funcionario f : lista){
		if(f.getSalario() >= salarioIni && f.getSalario() <= salarioFim){
			resultado.add(f);
		}
	}

	return resultado;
}
```

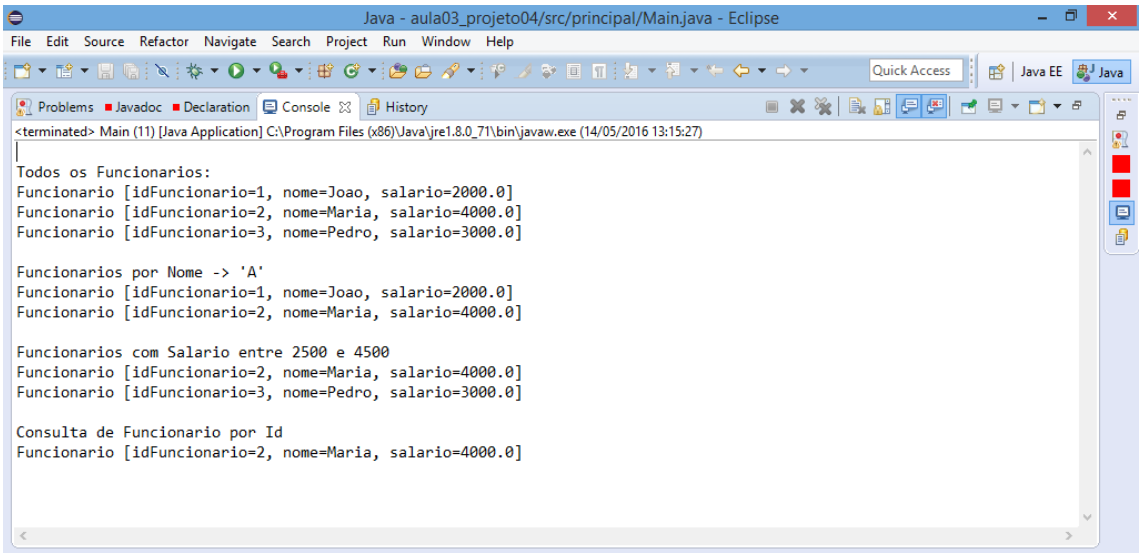
```
	@Override
	public List<Funcionario> getListagemFuncionarios() {
		//criando um ArrayList de funcionario e retornando-o...
		return new ArrayList<Funcionario>(lista);
	}

}
```

Testando no Main:

```
package principal;

import control.ControleFuncionario;
import entities.Funcionario;

public class Main {

	public static void main(String[] args) {

		Funcionario f1 = new Funcionario(1, "Joao",  2000.0);
		Funcionario f2 = new Funcionario(3, "Pedro", 3000.0);
		Funcionario f3 = new Funcionario(2, "Maria", 4000.0);

		ControleFuncionario c = new ControleFuncionario();

		c.adicionarFuncionario(f1);
		c.adicionarFuncionario(f2);
		c.adicionarFuncionario(f3);

		c.ordenarListagemFuncionarios();

		System.out.println("\nTodos os Funcionarios:");
		for(Funcionario f : c.getListagemFuncionarios()){
			System.out.println(f); //imprimindo..
		}

		System.out.println("\nFuncionarios por Nome -> 'A'");
		for(Funcionario f : c.obterPorNome("A")){
			System.out.println(f);
		}

		System.out.println("\nFuncionarios com Salario entre 2500 e 4500");
		for(Funcionario f : c.obterPorSalario(2500.0, 4500.0)){
			System.out.println(f);
		}
```

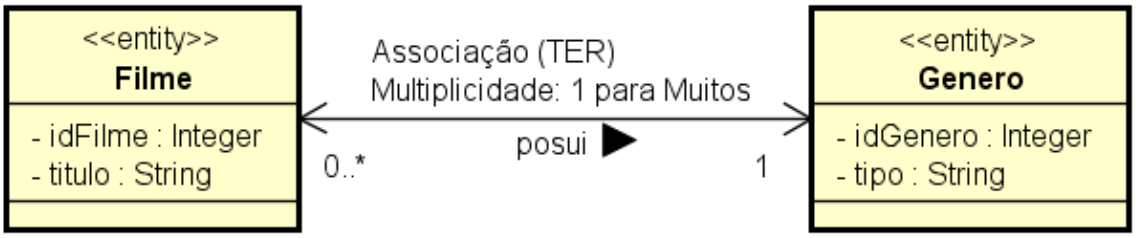
```
                //buscar por id..
                System.out.println("\nConsulta de Funcionario por Id");
                Funcionario f = c.obterPorId(2);

                if(f != null){
                        System.out.println(f); //toString()
                }
                else{
                        System.out.println("Funcionario não encontrado.");
                }
        }
}
```

Saida do programa:




```
Todos os Funcionarios:
Funcionario [idFuncionario=1, nome=Joao, salario=2000.0]
Funcionario [idFuncionario=2, nome=Maria, salario=4000.0]
Funcionario [idFuncionario=3, nome=Pedro, salario=3000.0]

Funcionarios por Nome -> 'A'
Funcionario [idFuncionario=1, nome=Joao, salario=2000.0]
Funcionario [idFuncionario=2, nome=Maria, salario=4000.0]

Funcionarios com Salario entre 2500 e 4500
Funcionario [idFuncionario=2, nome=Maria, salario=4000.0]
Funcionario [idFuncionario=3, nome=Pedro, salario=3000.0]

Consulta de Funcionario por Id
Funcionario [idFuncionario=2, nome=Maria, salario=4000.0]
```

Novo projeto:
Definindo uma modelagem de 1 para muitos




```
package entities;

public class Filme {

	private Integer idFilme;
	private String titulo;
	private Genero genero; //associação (TER-1)

	public Filme() {
		// TODO Auto-generated constructor stub
	}

	public Filme(Integer idFilme, String titulo) {
		this.idFilme = idFilme;
		this.titulo = titulo;
	}

	public Filme(Integer idFilme, String titulo, Genero genero) {
		this(idFilme, titulo);
		this.genero = genero;
	}

	public Integer getIdFilme() {
		return idFilme;
	}

	public void setIdFilme(Integer idFilme) {
		this.idFilme = idFilme;
	}

	public String getTitulo() {
		return titulo;
	}

	public void setTitulo(String titulo) {
		this.titulo = titulo;
	}
```

```
	public Genero getGenero() {
		return genero;
	}

	public void setGenero(Genero genero) {
		this.genero = genero;
	}

	@Override
	public String toString() {
		return "Filme [idFilme=" + idFilme + ", titulo="
				+ titulo + "]";
	}
}
```

```
package entities;

import java.util.List;

public class Genero {

	private Integer idGenero;
	private String tipo;
	private List<Filme> filmes; //Associação (muitos)

	public Genero() {
		// TODO Auto-generated constructor stub
	}

	public Genero(Integer idGenero, String tipo) {
		this.idGenero = idGenero;
		this.tipo = tipo;
	}

	public Genero(Integer idGenero, String tipo, List<Filme> filmes)
	{
		this(idGenero, tipo);
		this.filmes = filmes;
	}

	public Integer getIdGenero() {
		return idGenero;
	}

	public void setIdGenero(Integer idGenero) {
		this.idGenero = idGenero;
	}
```

```
	public String getTipo() {
		return tipo;
	}

	public void setTipo(String tipo) {
		this.tipo = tipo;
	}

	public List<Filme> getFilmes() {
		return filmes;
	}

	public void setFilmes(List<Filme> filmes) {
		this.filmes = filmes;
	}

	@Override
	public String toString() {
		return "Genero [idGenero=" + idGenero + ", tipo="
				+ tipo + "]";
	}
}
```

Criando uma classe para impressão dos dados do Genero e Filmes

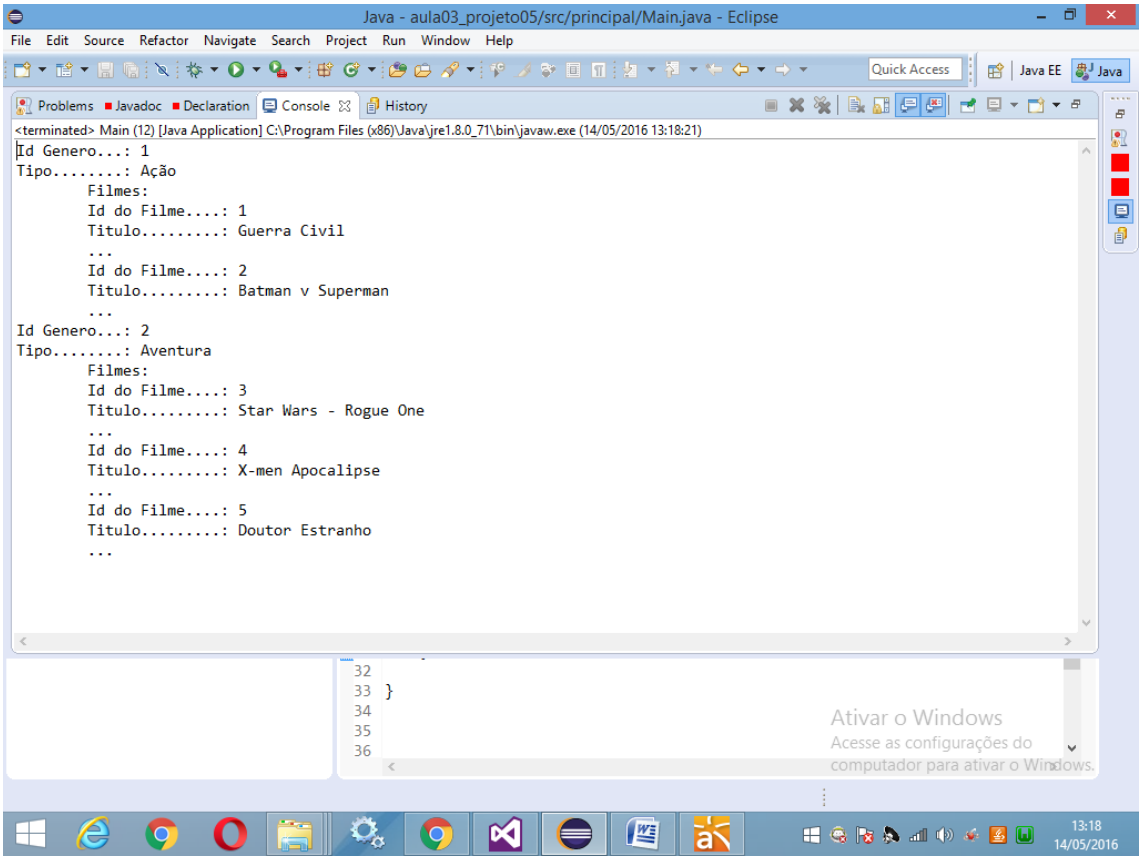
```
package output;

import entities.Filme;
import entities.Genero;

public class OutputGenero {

	//impressão no console..
	public void imprimir(Genero g){

		System.out.println("Id Genero...: " + g.getIdGenero());
		System.out.println("Tipo.....: " + g.getTipo());

		//verificando se o genero contem filmes..
		if(g.getFilmes() != null && g.getFilmes().size() > 0){

			System.out.println("\tFilmes:");
```

```
                        for(Filme f : g.getFilmes()){

                                System.out.println("\tId do Filme....: " + f.getIdFilme());
                                System.out.println("\tTitulo.....: " + f.getTitulo());
                                System.out.println("\t...");
                        }
                }
        }
}
```

Testando:

```
package principal;

import java.util.ArrayList;

import entities.Filme;
import entities.Genero;
import output.OutputGenero;

public class Main {

        public static void main(String[] args) {

                Genero g1 = new Genero(1, "Ação"); //instanciando a entidade..
                g1.setFilmes(new ArrayList<Filme>()); //instanciando a lista..

                Genero g2 = new Genero(2, "Aventura"); //instanciando a entidade..
                g2.setFilmes(new ArrayList<Filme>()); //instanciando a lista..

                //adicionar filmes aos generos..
                g1.getFilmes().add(new Filme(1, "Guerra Civil"));
                g1.getFilmes().add(new Filme(2, "Batman v Superman"));

                g2.getFilmes().add(new Filme(3, "Star Wars - Rogue One"));
                g2.getFilmes().add(new Filme(4, "X-men Apocalipse"));
                g2.getFilmes().add(new Filme(5, "Doutor Estranho"));

                //imprimindo..
                OutputGenero out = new OutputGenero();
                out.imprimir(g1); //imprimindo..
                out.imprimir(g2); //imprimindo..
        }

}
```

Saida do programa:




```
Id Genero...: 1
Tipo.....: Ação
	Filmes:
	Id do Filme....: 1
	Titulo.....: Guerra Civil
	...
	Id do Filme....: 2
	Titulo.....: Batman v Superman
	...
Id Genero...: 2
Tipo.....: Aventura
	Filmes:
	Id do Filme....: 3
	Titulo.....: Star Wars - Rogue One
	...
	Id do Filme....: 4
	Titulo.....: X-men Apocalipse
	...
	Id do Filme....: 5
	Titulo.....: Doutor Estranho
	...
```

Novo projeto:
Definindo uma entidade Cliente:

```
package entities;

public class Cliente implements Comparable<Cliente>{

	private Integer idCliente;
	private String nome;

	public Cliente() {
		// TODO Auto-generated constructor stub
	}

	public Cliente(Integer idCliente, String nome) {
		super();
		this.idCliente = idCliente;
		this.nome = nome;
	}

	public Integer getIdCliente() {
		return idCliente;
	}

	public void setIdCliente(Integer idCliente) {
		this.idCliente = idCliente;
	}

	public String getNome() {
		return nome;
	}

	public void setNome(String nome) {
		this.nome = nome;
	}

	@Override
	public String toString() {
		return "Cliente [idCliente=" + idCliente
			+ ", nome=" + nome + "]";
	}

	@Override
	public int compareTo(Cliente c) {
		return idCliente.compareTo(c.getIdCliente());
	}

}
```

Queue (Fila)

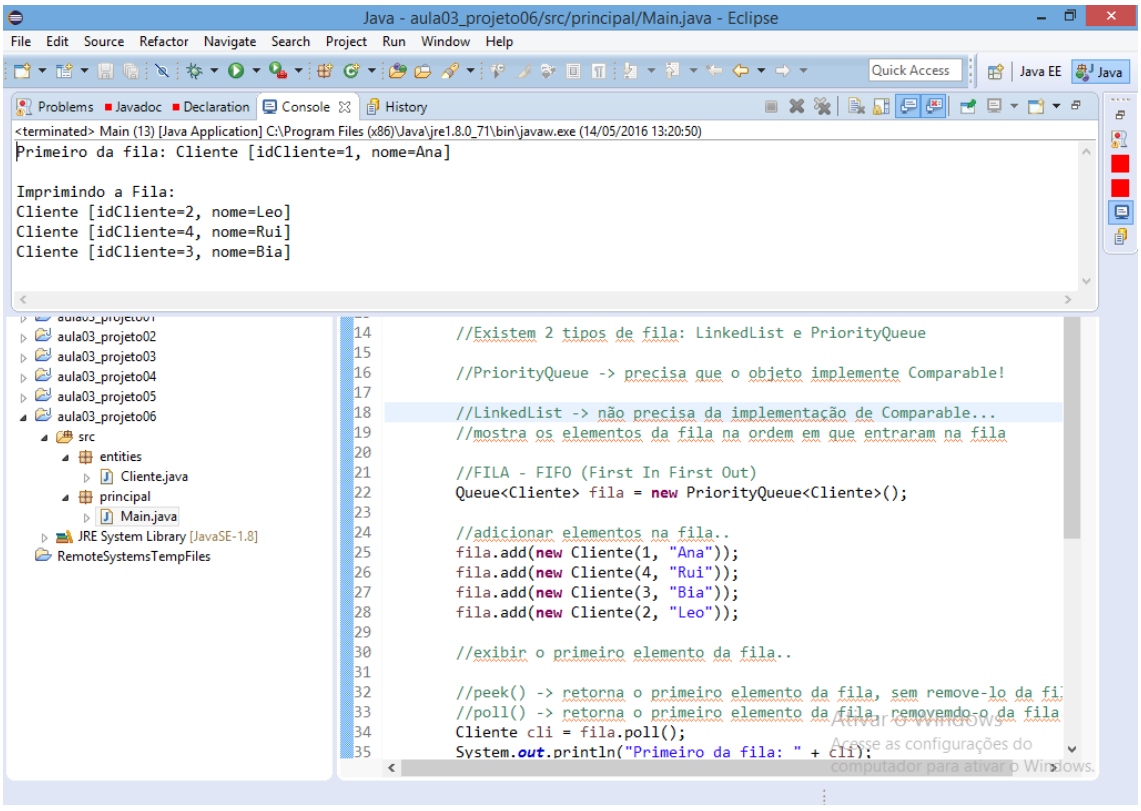
Nesta interface, o elemento é inserido na parte de trás da fila. Esta operação é chamada de **enfileiramento**. Este mesmo elemento sai a partir da frente da fila, operação chamada de **desenfileiramento**. Esse procedimento de entrada e saída recebe o nome de fila, ou **FIFO** (first-in first-out), ou seja, "primeiro a entrar, primeiro a sair".

Exemplo:

```
package principal;

import java.util.PriorityQueue;
import java.util.Queue;

import entities.Cliente;

public class Main {

	public static void main(String[] args) {

		//Queue -> manipulando fila (FIFO)

		//Existem 2 tipos de fila: LinkedList e PriorityQueue

		//PriorityQueue -> precisa que o objeto implemente Comparable!

		//LinkedList -> não precisa da implementação de Comparable...
		//mostra os elementos da fila na ordem em que entraram na fila

		//FILA - FIFO (First In First Out)
		Queue<Cliente> fila = new PriorityQueue<Cliente>();

		//adicionar elementos na fila..
		fila.add(new Cliente(1, "Ana"));
		fila.add(new Cliente(4, "Rui"));
		fila.add(new Cliente(3, "Bia"));
		fila.add(new Cliente(2, "Leo"));

		//exibir o primeiro elemento da fila..
```

```
                //peek() -> retorna o primeiro elemento da fila, sem remove-lo da fila..
                //poll() -> retorna o primeiro elemento da fila, removemdo-o da fila..
                Cliente cli = fila.poll();
                System.out.println("Primeiro da fila: " + cli);

                //imprimir a fila..
                System.out.println("\nImprimindo a Fila:");
                for(Cliente c : fila){
                        System.out.println(c);
                }
        }
}
```

Executando:




```
Primeiro da fila: Cliente [idCliente=1, nome=Ana]

Imprimindo a Fila:
Cliente [idCliente=2, nome=Leo]
Cliente [idCliente=4, nome=Rui]
Cliente [idCliente=3, nome=Bia]
```