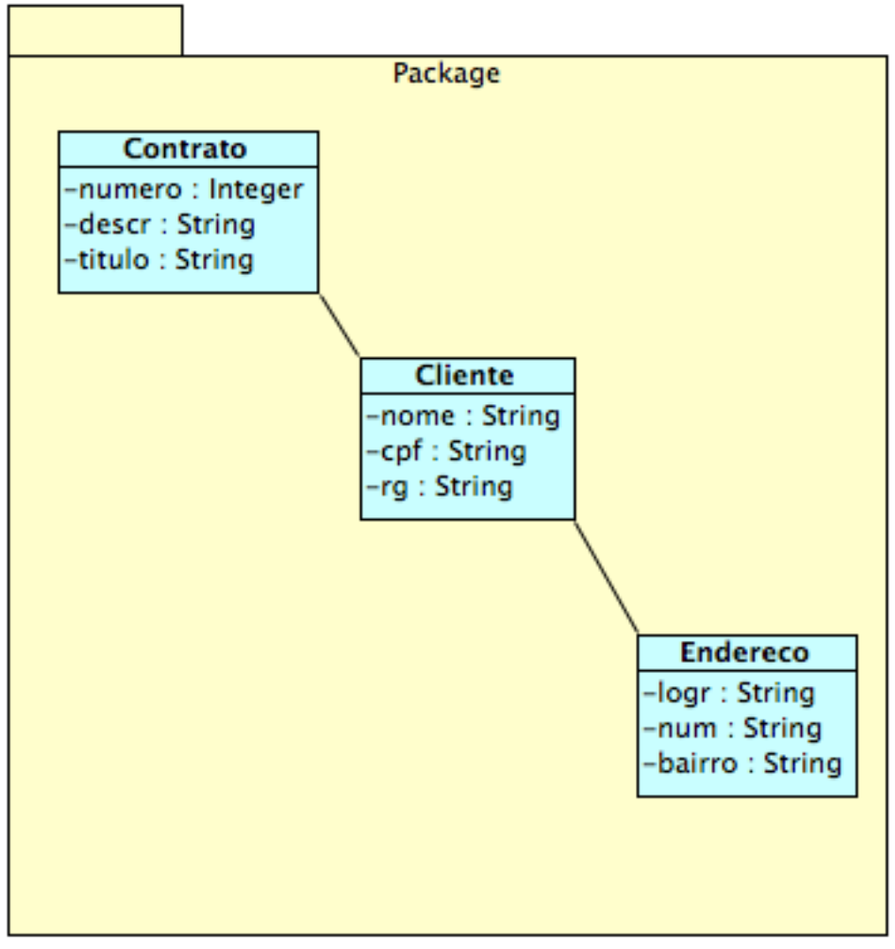
Orientação a Objetos

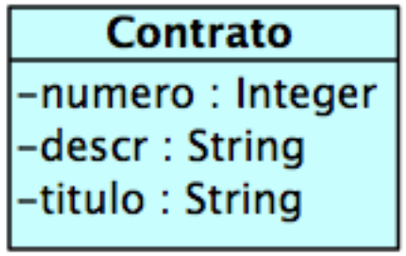
Agenda

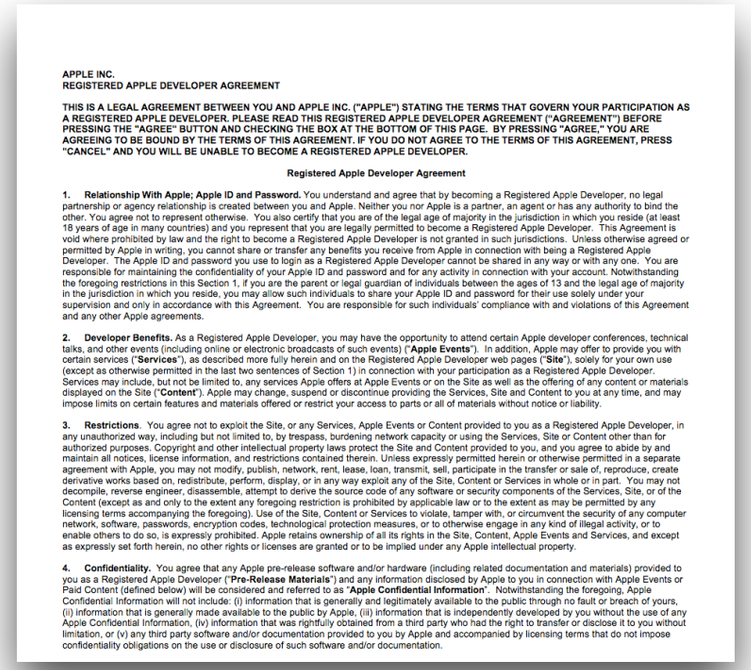
- Classes e Objetos
- Herança
- Agregação e Composição
- Inferência de Tipos, Tipos Genéricos, Métodos Genéricos e Tipos Alvo
- Interfaces
- Interfaces Funcionais
- Expressões Lambda
- Métodos referenciados



Classes e Objetos

Classes são estruturas de dados que representam abstrações de Objetos do mundo real.





APPLE INC.
REGISTERED APPLE DEVELOPER AGREEMENT

THIS IS A LEGAL AGREEMENT BETWEEN YOU AND APPLE INC. ("APPLE") STATING THE TERMS THAT GOVERN YOUR PARTICIPATION AS A REGISTERED APPLE DEVELOPER. PLEASE READ THIS REGISTERED APPLE DEVELOPER AGREEMENT ("AGREEMENT") BEFORE PRESSING THE "AGREE" BUTTON AND CHECKING THE BOX AT THE BOTTOM OF THIS PAGE. BY PRESSING "AGREE," YOU ARE AGREEING TO BE BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, PRESS "CANCEL" AND YOU WILL BE UNABLE TO BECOME A REGISTERED APPLE DEVELOPER.

Registered Apple Developer Agreement

1. Relationship With Apple; Apple ID and Password. You understand and agree that by becoming a Registered Apple Developer, no legal partnership or agency relationship is created between you and Apple. Neither you nor Apple is a partner, an agent or has any authority to bind the other. You agree not to represent otherwise. You also certify that you are of the legal age of majority in the jurisdiction in which you reside (at least 18 years of age in many countries) and you represent that you are legally permitted to become a Registered Apple Developer. This Agreement is void where prohibited by law and the right to become a Registered Apple Developer is not granted in such jurisdictions. Unless otherwise agreed or permitted by Apple in writing, you cannot share or transfer any benefits you receive from Apple in connection with being a Registered Apple Developer. The Apple ID and password you use to login as a Registered Apple Developer cannot be shared in any way or with any one. You are responsible for maintaining the confidentiality of your Apple ID and password and for any activity in connection with your account. Notwithstanding the foregoing restrictions in this Section 1, if you are the parent or legal guardian of individuals between the ages of 13 and the legal age of majority in the jurisdiction in which you reside, you may allow such individuals to share your Apple ID and password for their use solely under your supervision and only in accordance with this Agreement. You are responsible for such individuals' compliance with and violations of this Agreement and any other Apple agreements.

2. Developer Benefits. As a Registered Apple Developer, you may have the opportunity to attend certain Apple developer conferences, technical talks, and other events (including online or electronic broadcasts of such events) ("**Apple Events**"). In addition, Apple may offer to provide you with certain services ("**Services**"), as described more fully herein and on the Registered Apple Developer web pages ("**Site**"), solely for your own use (except as otherwise permitted in the last two sentences of Section 1) in connection with your participation as a Registered Apple Developer. Services may include, but not be limited to, any services Apple offers at Apple Events or on the Site as well as the offering of any content or materials displayed on the Site ("**Content**"). Apple may change, suspend or discontinue providing the Services, Site and Content to you at any time, and may impose limits on certain features and materials offered or restrict your access to parts or all of materials without notice or liability.

3. Restrictions. You agree not to exploit the Site, or any Services, Apple Events or Content provided to you as a Registered Apple Developer, in any unauthorized way, including but not limited to, by trespass, burdening network capacity or using the Services, Site or Content other than for authorized purposes. Copyright and other intellectual property laws protect the Site and Content provided to you, and you agree to abide by and maintain all notices, license information, and restrictions contained therein. Unless expressly permitted herein or otherwise permitted in a separate agreement with Apple, you may not modify, publish, network, rent, lease, loan, transmit, sell, participate in the transfer or sale of, reproduce, create derivative works based on, redistribute, perform, display, or in any way exploit any of the Site, Content or Services in whole or in part. You may not decompile, reverse engineer, disassemble, attempt to derive the source code of any software or security components of the Services, Site, or of the Content (except as and only to the extent any foregoing restriction is prohibited by applicable law or to the extent as may be permitted by any licensing terms accompanying the foregoing). Use of the Site, Content or Services to violate, tamper with, or circumvent the security of any computer network, software, passwords, encryption codes, technological protection measures, or to otherwise engage in any kind of illegal activity, or to enable others to do so, is expressly prohibited. Apple retains ownership of all its rights in the Site, Content, Apple Events and Services, and except as expressly set forth herein, no other rights or licenses are granted or to be implied under any Apple intellectual property.

4. Confidentiality. You agree that any Apple pre-release software and/or hardware (including related documentation and materials) provided to you as a Registered Apple Developer ("**Pre-Release Materials**") and any information disclosed by Apple to you in connection with Apple Events or Paid Content (defined below) will be considered and referred to as "**Apple Confidential Information**". Notwithstanding the foregoing, Apple Confidential Information will not include: (i) information that is generally and legitimately available to the public through no fault or breach of yours, (ii) information that is generally made available to the public by Apple, (iii) information that is independently developed by you without the use of any Apple Confidential Information, (iv) information that was rightfully obtained from a third party who had the right to transfer or disclose it to you without limitation, or (v) any third party software and/or documentation provided to you by Apple and accompanied by licensing terms that do not impose confidentiality obligations on the use or disclosure of such software and/or documentation.

Estas abstrações são representadas na forma de atributos, necessários ao armazenamento da informações que serão utilizadas nas aplicações.

Classes e Objetos

Uma Classe é declarada contento um **nome**, seus **atributos privados** e os **métodos públicos**.

O nome da classe define um novo **Tipo de Dado** seus métodos permitem controlar a leitura e a gravação em seus atributos.

```java
public class Contrato {
  private String numero;
  private String descr;
  private String titulo;

  public String getNumero() {
      return numero;
  }

  public void setNumero(String numero) {
      this.numero = numero;
  }

  public String getDescr() {
      return descr;
  }

  public void setDescr(String descr) {
      this.descr = descr;
  }

  public String getTitulo() {
      return titulo;
  }

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

Classes e Objetos

Uma Classe pode ser **publica** ou **privada**, **abstrata** ou **final**.

Public permite sua utilização em qualquer local, **private** oculta sua implementação.

Abstract exige a complementação se sua implementação, **final** impede sua extensão (herança).

```
private class Produto { }

public class Processo { }

public abstract class Figura {
    public abstract void exibir();
}

public final class String { }
```

Atributos e métodos

Os atributos representam as informações que fazem parte das classes. Os métodos representam as ações ou mensagens que podemos requisitar ou enviar aos objetos.

Eles podem ser **privados**, **protegidos** e **públicos**.

A estas características permitem controlar o nível de encapsulamento destes atributos e métodos.

```
public abstract class Figura {
  private Integer coordenada;
  public String nome;
  protected String tipo;

  public void move(int x, int y) { }

  public abstract void desenha();
}
```

Atributos e métodos

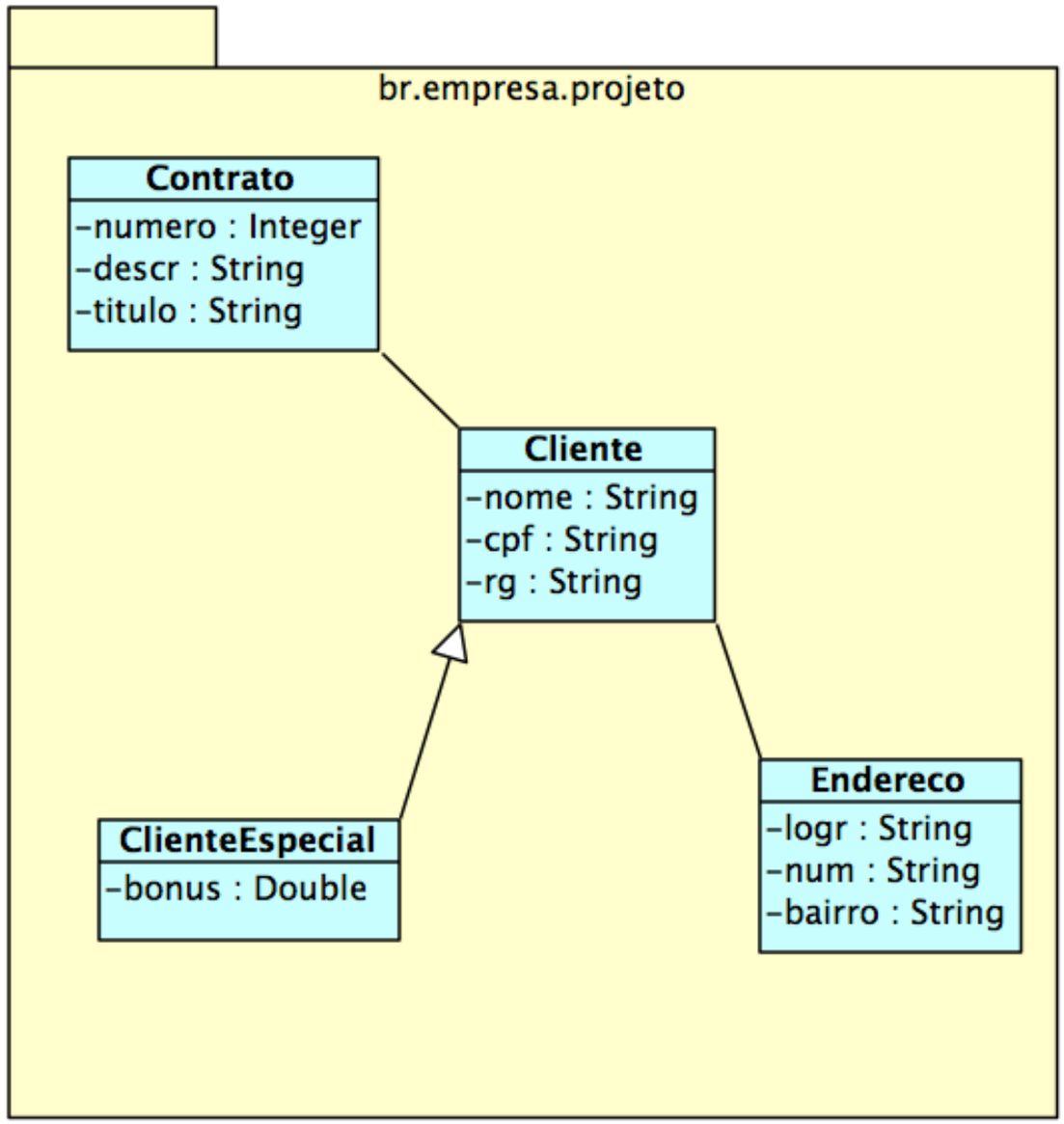
Podemos definir a forma de utilização dos atributos e métodos com os modificadores **static**, **final** e **transient**.

Com **static** fixamos os atributos e métodos na classe e não no objeto, com **final** criamos atributos constantes e impedimos a substituição de métodos, quando utilizando herança, por fim **transient** declara atributos temporários.

```
class Circulo extends Figura {
  private final Integer posicao = 100;
  private transient String temp;
  private static Integer tamanho;

  @Override
  public void desenha() { }
}
```

Relacionamento entre Classes de Objetos

Os relacionamentos e interações existentes entre as classes de objetos são representadas na forma de **heranças**, **agregações**, **composições** e **referências**.



Herança

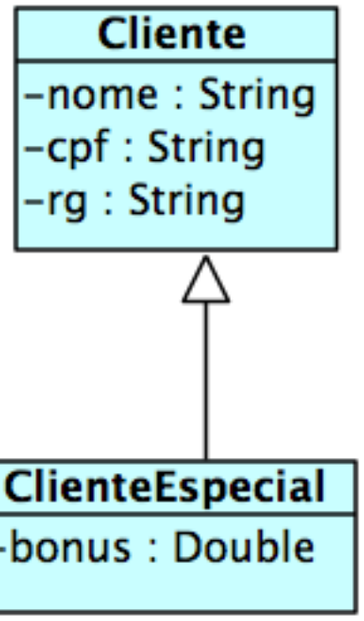
A Herança permite a especialização de uma Classe, desta forma poderemos ampliar as implementações de algoritmos além do que já foi declarado na Super Classe.

Utilizamos a palavra reservada **extends** para criar herança entre classes.

Usamos **this** para referenciar o próprio objeto e **super** para referenciar o objeto pai.

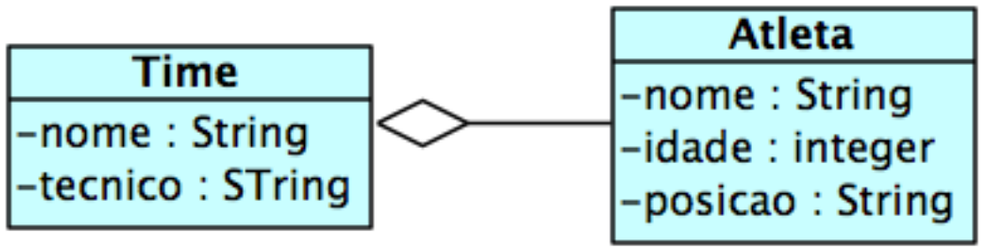
```java
public class Cliente {
  private String nome;
  private String cpf;
  private String rg;

  public String getNome() {..

  public void setNome(String nome) {..

  public String getCpf() {..

  public void setCpf(String cpf) {..

  public String getRg() {..

  public void setRg(String rg) {..
}
```

```java
public class ClienteEspecial extends Cliente {
  private Double bonus;

  public Double getBonus() {..

  public void setBonus(Double bonus) {..
}
```

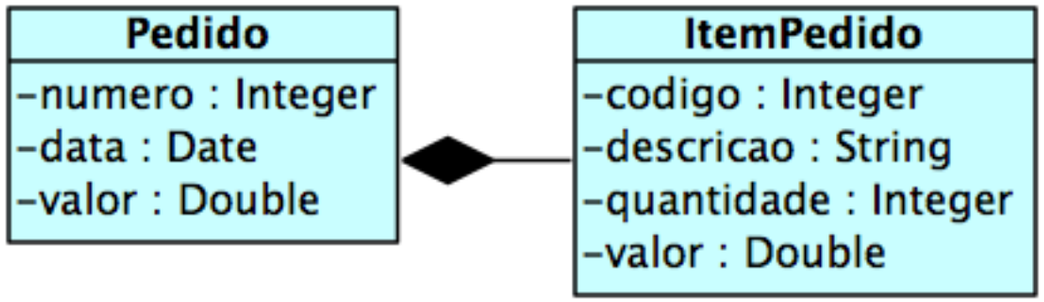


Agregação e composição

Chamamos de agregação ou composição, quando uma classe necessita de outra para sua construção.



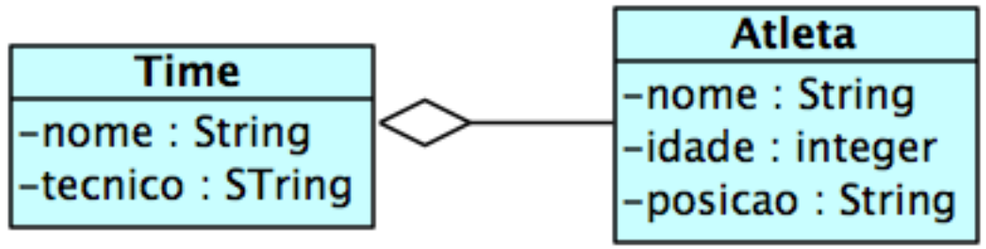
Se a relação entre as classes é mais fraca chamamos de **agregação**, se for mais forte chamamos de **composição**.



Agregação

Na agregação o Time pode ter zero ou mais Atletas.

Esta relação é 1 para 0 ou N, assim temos uma coleção de Atletas como atributo no Time.



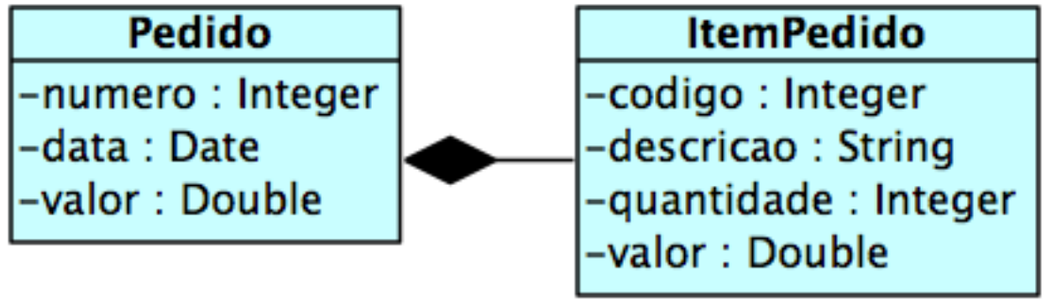

```
class Time {
    private String nome;
    private String tecnico;
    private Set<Atleta> oAtleta;
}
```

```
class Atleta {
    private String nome;
    private Integer idade;
    private String posicao;
}
```

Composição

Na composição o Pedido tem ter um ou mais Itens de Pedido.

Esta relação é 1 para N, assim temos uma coleção de ItemPedido como atributo no Pedido.

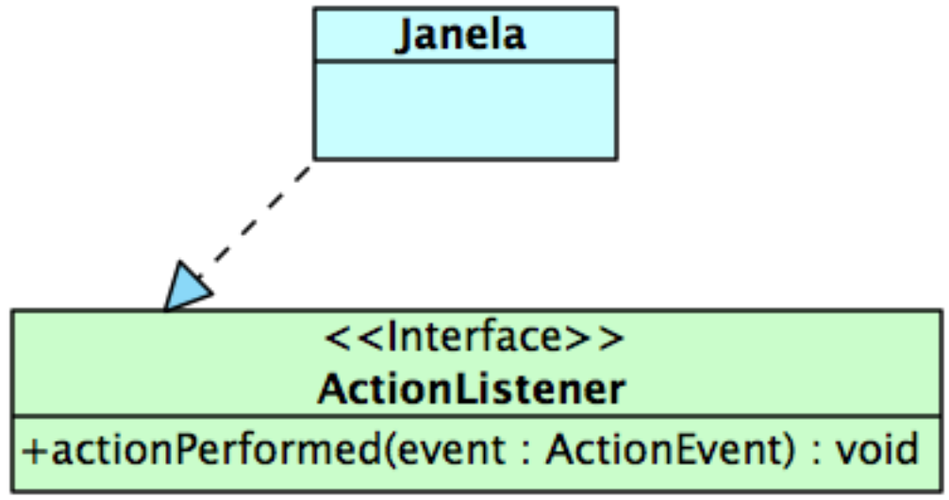



```
class Pedido {
    private Integer numero;
    private Date data;
    private Double valor;
    private Set<ItemPedido> itens;
}

class ItemPedido {
    private Integer codigo;
    private String descricao;
    private Integer quantidade;
    private Double valor;
}
```

Interfaces

As **Interfaces** representam contratos que as classe devem implementar.

Uma classe pode implementar mais de uma interface.




```
class Janela implements ActionListener {
  @Override
  public void actionPerformed(ActionEvent e) {
  }
}
```

Interfaces

Uma classe pode implementar uma interface de várias maneiras.

As duas últimas formas utilizam expressões Lambda em sua implementação.

```
class FazAlgo implements Comparator<Cliente> {
  @Override
  public int compare(Cliente o1, Cliente o2) {
    return o1.getNome().compareTo(o2.getNome());
  }
}

Arrays.sort(lista, new FazAlgo());
```

```
Arrays.sort(lista, new Comparator<Cliente>() {
  @Override
  public int compare(Cliente o1, Cliente o2) {
    return o1.getNome().compareTo(o2.getNome());
  }
});
```

```
Arrays.sort(lista, (Cliente o1, Cliente o2) ->
     o1.getNome().compareTo(o2.getNome())
 );
```

```
Arrays.sort(lista, (o1, o2) ->
     o1.getNome().compareTo(o2.getNome())
 );
```

Interfaces Funcionais

As interfaces funcionais fornecem tipos de retorno para as expressões lambda e referências de métodos.

Cada interface funcional tem um método abstrato simples, chamado de método funcional para essa interface, para a qual o parâmetro de expressão lambda e tipos de retorno são combinados ou adaptados.

```
@FunctionalInterface
public interface Runnable {
  public abstract void run();
}
```

Tipos Genéricos

Um Tipo genérico é uma classe ou uma interface que é parametrizada por um tipo.

Este tipo pode ser utilizado como tipo de argumentos ou de retorno em métodos e atributos.

```
class Box<T> {
  private T item;

  public void set(T item) {
      this.item = item;
  }

    public T get() {
        return item;
    }
}
```

Expressões Lambda

A expressão Lambda representa funções anônimas.

São compostas por um ou mais argumentos, o operador Lambda **->** , e o bloco da função.

As expressões Lambda permitem que um bloco de lógica seja passado como argumento na chamada a um método.

```
Cliente[] lista = {
    new Cliente()
        .setNome("Fulano")
        .setEmail("fulano@gmail.com")
        .setIdade(22),
    new Cliente()
        .setNome("Beltrano")
        .setEmail("beltrano@bol.com.br")
        .setIdade(32) };

Arrays.sort(lista,
    (o1, o2) -> o1.getNome().compareTo(o2.getNome()));

Arrays.stream(lista)
    .filter(e -> e.getIdade() > 20)
    .forEach((e) -> System.out.println(e));
```

Métodos referenciados

Referência a métodos se semelham a atalhos para algumas formas de expressões Lambda.

Método referenciado	Expressão Lambda equivalente
`String::valueOf`	`x -> String.valueOf(x)`
`Object::toString`	`x -> x.toString()`
`x::toString`	`() -> x.toString()`
`ArrayList::new`	`() -> new ArrayList<>()`

Referências

- http://www.oracle.com/events/us/en/java8/index.html
- http://www.techempower.com/blog/2013/03/26/everything-about-java-8/
- http://docs.oracle.com/javase/8/docs/api/java/util/function/package-summary.html
- http://docs.oracle.com/javase/tutorial/index.html
- http://www.oracle.com/technetwork/java/javase/8-whats-new-2157071.html