Module com.example.supertrunfo

`module com.example.supertrunfo`

Fornece a aplicação principal e classes relacionadas.

Requisitos:

- `javafx.controls` - Biblioteca JavaFX para controles gráficos.
- `javafx.fxml` - Biblioteca JavaFX para carregamento e manipulação de arquivos FXML.
- `java.sql` - Biblioteca Java para acesso a banco de dados SQL.
- `jbcrypt` - Biblioteca para criptografia de senhas utilizando o algoritmo BCrypt.
- `java.desktop` - Pacote Java para recursos de área de trabalho, como integração com o sistema de arquivos.

Este módulo fornece a aplicação principal SuperTrunfo e outras classes relacionadas necessárias para o funcionamento do jogo.

Packages

All Packages Exports Opens Concealed

Package	Exported To Modules	Opened To Modules	Description
poo.trabalhofinal.supertrunfo	All Modules	javafx.fxml	
poo.trabalhofinal.supertrunfo.classes	None	None	
poo.trabalhofinal.supertrunfo.classes.carta	All Modules	None	
poo.trabalhofinal.supertrunfo.classes.exce	None	None	
poo.trabalhofinal.supertrunfo.classes.inter	None	None	
poo.trabalhofinal.supertrunfo.classes.utils	None	None	
poo.trabalhofinal.supertrunfo.gui	All Modules	javafx.fxml	
poo.trabalhofinal.supertrunfo.gui.controlle	All Modules	javafx.fxml	

Indirect Exports

From	Packages
java.base	com.sun.crypto.provider com.sun.security.ntlm java.io java.lang java.lang.annotation java.lang.constant java.lang.foreign java.lang.invoke java.lang.module java.lang.ref java.lang.reflect java.lang.runtime java.math java.net java.net.spi java.nio java.nio.channels java.nio.channels.spi java.nio.charset java.nio.charset.spi java.nio.file java.nio.file.attribute java.nio.file.spi java.security java.security.cert java.security.interfaces java.security.spec java.text java.text.spi java.time java.time.chrono java.time.format java.time.temporal java.time.zone java.util java.util.concurrent java.util.concurrent.atomic java.util.concurrent.locks

	java.util.function java.util.jar java.util.random java.util.regex java.util.spi java.util.stream java.util.zip javax.crypto javax.crypto.interfaces javax.crypto.spec javax.net javax.net.ssl javax.security.auth javax.security.auth.callback javax.security.auth.login javax.security.auth.spi javax.security.auth.x500 javax.security.cert jdk.internal.access jdk.internal.event jdk.internal.foreign jdk.internal.io jdk.internal.javac jdk.internal.jimage jdk.internal.jimage.decompressor jdk.internal.jmod jdk.internal.loader jdk.internal.logger jdk.internal.misc jdk.internal.module jdk.internal.org.objectweb.asm jdk.internal.org.objectweb.asm.commons jdk.internal.org.objectweb.asm.tree jdk.internal.org.objectweb.asm.util jdk.internal.org.xml.sax jdk.internal.org.xml.sax.helpers jdk.internal.perf jdk.internal.platform jdk.internal.ref jdk.internal.reflect jdk.internal.util.jar jdk.internal.util.random jdk.internal.util.xml jdk.internal.util.xml.impl jdk.internal.vm jdk.internal.vm.annotation jdk.internal.vm.vector sun.invoke.util sun.net sun.net.dns sun.net.ext sun.net.util sun.net.www sun.net.www.protocol.http sun.nio.ch sun.nio.cs sun.nio.fs sun.reflect.annotation sun.reflect.generics.reflectiveObjects sun.reflect.misc sun.security.action sun.security.internal.interfaces sun.security.internal.spec sun.security.jca sun.security.pkcs sun.security.provider sun.security.provider.certpath sun.security.rsa sun.security.timestamp sun.security.tools sun.security.util sun.security.util.math sun.security.util.math.intpoly sun.security.validator sun.security.x509 sun.util.cldr sun.util.locale.provider sun.util.logging sun.util.resources
java.desktop	java.applet java.awt java.awt.color java.awt.desktop java.awt.dnd java.awt.dnd.peer java.awt.event java.awt.font java.awt.geom java.awt.im java.awt.im.spi java.awt.image java.awt.image.renderable java.awt.print java.beans java.beans.beancontext javax.accessibility javax.imageio javax.imageio.event javax.imageio.metadata javax.imageio.plugins.bmp javax.imageio.plugins.jpeg javax.imageio.plugins.tiff javax.imageio.spi javax.imageio.stream javax.print javax.print.attribute javax.print.attribute.standard javax.print.event javax.sound.midi javax.sound.midi.spi javax.sound.sampled javax.sound.sampled.spi javax.swing javax.swing.border javax.swing.colorchooser javax.swing.event javax.swing.filechooser javax.swing.plaf javax.swing.plaf.basic javax.swing.plaf.metal javax.swing.plaf.multi javax.swing.plaf.nimbus javax.swing.plaf.synth javax.swing.table javax.swing.text javax.swing.text.html javax.swing.text.html.parser javax.swing.text.rtf javax.swing.tree javax.swing.undo sun.awt sun.awt.dnd sun.swing
java.sql	java.sql javax.sql
javafx.controls	com.sun.javafx.scene.control com.sun.javafx.scene.control.behavior com.sun.javafx.scene.control.inputmap com.sun.javafx.scene.control.skin javafx.scene.chart javafx.scene.control javafx.scene.control.cell javafx.scene.control.skin
javafx.fxml	javafx.fxml
jbcrypt	org.mindrot.jbcrypt

Indirect Opens

From	Packages
java.desktop	com.sun.java.swing.plaf.windows javax.swing.plaf.basic

Modules

Requires

Modifier	Module	Description
	java.base	
	java.desktop	
	java.sql	
	javafx.controls	
	javafx.fxml	
	jbcrypt	

All Classes and Interfaces

All Classes and Interfaces | Interfaces | Classes | Enum Classes | Exception Classes

Class	Description
CadastroCartaController	Classe CadastroCartaController
CadastroUsuarioController	Classe CadastroUsuarioController
Carta	Classe Carta
CartasRepository<T>	Interface CartasRepository
CartasRepositoryImpl<T>	Classe CartasRepositoryImpl
Classificacao	Enum Classificação
DBUtils	Classe DBUtils
Gato	Classe Gato
HelloApplication	Classe HelloApplication
InformacaoInvalidaException	Classe de Exceção para informações inválidas
Jogador<T>	Classe Jogador
JogadoresRepository<T>	Interface JogoRepository
JogadoresRepositoryImpl<T>	Classe JogadoresRepositoryImpl
Jogo<T>	Classe Jogo
JogoController	Classe JogoController
JogoException	Classe de Exceção para Jogo
LinguagensProgramacao	Classe LinguagensProgramacao
LoginContoller	Classe LoginContoller
MenuController	Classe MenuController
OpcaoController	Classe OpcaoController
Personagem	Classe Personagem
TelaInicialController	Classe TelaInicialController
UsuarioNaoEncontradoException	Classe de Exceção para Usuário não encontrado
Util	Classe Util
VencedorController	Classe VencedorController
VerCartasController	Classe VerCartasController

Hierarchy For All Packages

Package Hierarchies:
poo.trabalhofinal.supertrunfo, poo.trabalhofinal.supertrunfo.classes, poo.trabalhofinal.supertrunfo.classes.cartas, poo.trabalhofinal.supertrunfo.classes.exceptions, poo.trabalhofinal.supertrunfo.classes.interfaces, poo.trabalhofinal.supertrunfo.classes.utils, poo.trabalhofinal.supertrunfo.gui, poo.trabalhofinal.supertrunfo.gui.controllers

Class Hierarchy

- java.lang.**Object**
  - javafx.application.Application
    - poo.trabalhofinal.supertrunfo.**HelloApplication**
  - poo.trabalhofinal.supertrunfo.gui.controllers.**CadastroCartaController** (implements javafx.fxml.Initializable)
  - poo.trabalhofinal.supertrunfo.gui.controllers.**CadastroUsuarioController** (implements javafx.fxml.Initializable)
  - poo.trabalhofinal.supertrunfo.classes.cartas.**Carta**
    - poo.trabalhofinal.supertrunfo.classes.cartas.**Gato**
    - poo.trabalhofinal.supertrunfo.classes.cartas.**LinguagensProgramacao**
    - poo.trabalhofinal.supertrunfo.classes.cartas.**Personagem**
  - poo.trabalhofinal.supertrunfo.classes.interfaces.**CartasRepositoryImpl**<T> (implements poo.trabalhofinal.supertrunfo.classes.interfaces.CartasRepository<T>)
  - poo.trabalhofinal.supertrunfo.gui.**DBUtils**
  - poo.trabalhofinal.supertrunfo.classes.**Jogador**<T>
  - poo.trabalhofinal.supertrunfo.classes.interfaces.**JogadoresRepositoryImpl**<T> (implements poo.trabalhofinal.supertrunfo.classes.interfaces.JogadoresRepository<T>)
  - poo.trabalhofinal.supertrunfo.classes.**Jogo**<T>
  - poo.trabalhofinal.supertrunfo.gui.controllers.**JogoController** (implements javafx.fxml.Initializable)
  - poo.trabalhofinal.supertrunfo.gui.controllers.**LoginContoller** (implements javafx.fxml.Initializable)
  - poo.trabalhofinal.supertrunfo.gui.controllers.**MenuController** (implements javafx.fxml.Initializable)
  - poo.trabalhofinal.supertrunfo.gui.controllers.**OpcaoController** (implements javafx.fxml.Initializable)
  - poo.trabalhofinal.supertrunfo.gui.controllers.**TelaInicialController** (implements javafx.fxml.Initializable)
  - java.lang.**Throwable** (implements java.io.Serializable)
    - java.lang.**Exception**
      - poo.trabalhofinal.supertrunfo.classes.exceptions.**InformacaoInvalidaException**
      - poo.trabalhofinal.supertrunfo.classes.exceptions.**JogoException**
      - poo.trabalhofinal.supertrunfo.classes.exceptions.**UsuarioNaoEncontradoException**
  - poo.trabalhofinal.supertrunfo.classes.utils.**Util**
  - poo.trabalhofinal.supertrunfo.gui.controllers.**VencedorController**
  - poo.trabalhofinal.supertrunfo.gui.controllers.**VerCartasController** (implements javafx.fxml.Initializable)

Interface Hierarchy

- poo.trabalhofinal.supertrunfo.classes.interfaces.**CartasRepository**<T>
- poo.trabalhofinal.supertrunfo.classes.interfaces.**JogadoresRepository**<T>

Enum Class Hierarchy

- java.lang.**Object**
  - java.lang.**Enum** <E> (implements java.lang.Comparable <T>, java.lang.constant.Constable , java.io.Serializable)
    - poo.trabalhofinal.supertrunfo.classes.cartas.**Classificacao**

Module com.example.supertrunfo

`module com.example.supertrunfo`

Fornece a aplicação principal e classes relacionadas.

Requisitos:

- `javafx.controls` - Biblioteca JavaFX para controles gráficos.
- `javafx.fxml` - Biblioteca JavaFX para carregamento e manipulação de arquivos FXML.
- `java.sql` - Biblioteca Java para acesso a banco de dados SQL.
- `jbcrypt` - Biblioteca para criptografia de senhas utilizando o algoritmo BCrypt.
- `java.desktop` - Pacote Java para recursos de área de trabalho, como integração com o sistema de arquivos.

Este módulo fornece a aplicação principal SuperTrunfo e outras classes relacionadas necessárias para o funcionamento do jogo.

Packages

All Packages Exports Opens Concealed

Package	Exported To Modules	Opened To Modules	Description
poo.trabalhofinal.supertrunfo	All Modules	javafx.fxml	
poo.trabalhofinal.supertrunfo.classes	None	None	
poo.trabalhofinal.supertrunfo.classes.carta	All Modules	None	
poo.trabalhofinal.supertrunfo.classes.exce	None	None	
poo.trabalhofinal.supertrunfo.classes.inter	None	None	
poo.trabalhofinal.supertrunfo.classes.utils	None	None	
poo.trabalhofinal.supertrunfo.gui	All Modules	javafx.fxml	
poo.trabalhofinal.supertrunfo.gui.controlle	All Modules	javafx.fxml	

Indirect Exports

From	Packages
java.base	com.sun.crypto.provider com.sun.security.ntlm java.io java.lang java.lang.annotation java.lang.constant java.lang.foreign java.lang.invoke java.lang.module java.lang.ref java.lang.reflect java.lang.runtime java.math java.net java.net.spi java.nio java.nio.channels java.nio.channels.spi java.nio.charset java.nio.charset.spi java.nio.file java.nio.file.attribute java.nio.file.spi java.security java.security.cert java.security.interfaces java.security.spec java.text java.text.spi java.time java.time.chrono java.time.format java.time.temporal java.time.zone java.util java.util.concurrent java.util.concurrent.atomic java.util.concurrent.locks

	java.util.function java.util.jar java.util.random java.util.regex java.util.spi java.util.stream java.util.zip javax.crypto javax.crypto.interfaces javax.crypto.spec javax.net javax.net.ssl javax.security.auth javax.security.auth.callback javax.security.auth.login javax.security.auth.spi javax.security.auth.x500 javax.security.cert jdk.internal.access jdk.internal.event jdk.internal.foreign jdk.internal.io jdk.internal.javac jdk.internal.jimage jdk.internal.jimage.decompressor jdk.internal.jmod jdk.internal.loader jdk.internal.logger jdk.internal.misc jdk.internal.module jdk.internal.org.objectweb.asm jdk.internal.org.objectweb.asm.commons jdk.internal.org.objectweb.asm.tree jdk.internal.org.objectweb.asm.util jdk.internal.org.xml.sax jdk.internal.org.xml.sax.helpers jdk.internal.perf jdk.internal.platform jdk.internal.ref jdk.internal.reflect jdk.internal.util.jar jdk.internal.util.random jdk.internal.util.xml jdk.internal.util.xml.impl jdk.internal.vm jdk.internal.vm.annotation jdk.internal.vm.vector sun.invoke.util sun.net sun.net.dns sun.net.ext sun.net.util sun.net.www sun.net.www.protocol.http sun.nio.ch sun.nio.cs sun.nio.fs sun.reflect.annotation sun.reflect.generics.reflectiveObjects sun.reflect.misc sun.security.action sun.security.internal.interfaces sun.security.internal.spec sun.security.jca sun.security.pkcs sun.security.provider sun.security.provider.certpath sun.security.rsa sun.security.timestamp sun.security.tools sun.security.util sun.security.util.math sun.security.util.math.intpoly sun.security.validator sun.security.x509 sun.util.cldr sun.util.locale.provider sun.util.logging sun.util.resources
java.desktop	java.applet java.awt java.awt.color java.awt.desktop java.awt.dnd java.awt.dnd.peer java.awt.event java.awt.font java.awt.geom java.awt.im java.awt.im.spi java.awt.image java.awt.image.renderable java.awt.print java.beans java.beans.beancontext javax.accessibility javax.imageio javax.imageio.event javax.imageio.metadata javax.imageio.plugins.bmp javax.imageio.plugins.jpeg javax.imageio.plugins.tiff javax.imageio.spi javax.imageio.stream javax.print javax.print.attribute javax.print.attribute.standard javax.print.event javax.sound.midi javax.sound.midi.spi javax.sound.sampled javax.sound.sampled.spi javax.swing javax.swing.border javax.swing.colorchooser javax.swing.event javax.swing.filechooser javax.swing.plaf javax.swing.plaf.basic javax.swing.plaf.metal javax.swing.plaf.multi javax.swing.plaf.nimbus javax.swing.plaf.synth javax.swing.table javax.swing.text javax.swing.text.html javax.swing.text.html.parser javax.swing.text.rtf javax.swing.tree javax.swing.undo sun.awt sun.awt.dnd sun.swing
java.sql	java.sql javax.sql
javafx.controls	com.sun.javafx.scene.control com.sun.javafx.scene.control.behavior com.sun.javafx.scene.control.inputmap com.sun.javafx.scene.control.skin javafx.scene.chart javafx.scene.control javafx.scene.control.cell javafx.scene.control.skin
javafx.fxml	javafx.fxml
jbcrypt	org.mindrot.jbcrypt

Indirect Opens

From	Packages
java.desktop	com.sun.java.swing.plaf.windows javax.swing.plaf.basic

Modules

Requires

Modifier	Module	Description
	java.base	
	java.desktop	
	java.sql	
	javafx.controls	
	javafx.fxml	
	jbcrypt	

Module com.example.supertrunfo

Package poo.trabalhofinal.supertrunfo

Class HelloApplication

java.lang.Object
    javafx.application.Application
        poo.trabalhofinal.supertrunfo.HelloApplication

```
public class HelloApplication
extends javafx.application.Application
```

Classe HelloApplication

Classe responsável por inicializar os elevemtos do JavaFX. É o ponto de entrada para os aplicativos FXML.

Version:

1.0

Author:

Lucas Furriel Rodrigues, Júlia Rodrigues Marques do Nascimento

Nested Class Summary

Nested classes/interfaces inherited from class javafx.application.Application

`javafx.application.Application.Parameters`

Field Summary

Fields inherited from class javafx.application.Application

`STYLESHEET_CASPIAN, STYLESHEET_MODENA`

Constructor Summary

Constructors

Constructor	Description
`HelloApplication()`	

Method Summary

All Methods | Static Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
static void	main(String [] args)	Método main que fica responsável por inicializar a aplicação JavaFX.
void	start(javafx.stage.Stage stage)	Classe responsável por criar um objeto FXMLLoader para carregar o arquivo FXML da tela inicial.

Methods inherited from class javafx.application.Application

getHostServices, getParameters, getUserAgentStylesheet, init, launch, launch, notifyPreloader, setUserAgentStylesheet, stop

Methods inherited from class java.lang.Object

clone , equals , finalize , getClass , hashCode , notify , notifyAll , toString , wait , wait , wait

Constructor Details

HelloApplication

```
public HelloApplication()
```

Method Details

start

```
public void start(javafx.stage.Stage stage)
           throws IOException
```

Classe responsável por criar um objeto FXMLLoader para carregar o arquivo FXML da tela inicial.

Carrega o arquivo FXML e cria uma cena com tamanho definido, que não é redimensionável.

Coloca um ícone em todas as janelas.

Responsável por mostrar as cenas na aplicação.

Specified by:

`start` in class `javafx.application.Application`

Parameters:

`stage` - objeto Stage como argumento -> representa a janela principal do aplicativo.

Throws:

`IOException` - Se houver erro ao inicializar o JavaFX

main

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

Método main que fica responsável por inicializar a aplicação JavaFX.

Parameters:

`args` - ('String')

Module com.example.supertrunfo
Package poo.trabalhofinal.supertrunfo.classes

Class Jogo<T>

java.lang.Object
    poo.trabalhofinal.supertrunfo.classes.Jogo<T>

Type Parameters:

T - Generics que representa o tipo de baralho que está sendo utilizado no jogo.

```
public class Jogo<T>
extends Object
```

Classe Jogo

Classe responsável por inicializar o jogo. Esta classe representa o jogo que os usuários estão jogando.

Field Summary

Fields

Modifier and Type	Field	Description
`private ArrayList <T>`	`baralho`	ArrayList baralho: lista de cartas de um tipo de baralho específico.
`private Jogador<T>`	`jogadorA`	Jogador jogadorA: primeiro jogador logado para o jogo.
`private Jogador<T>`	`jogadorB`	Jogador jogadorB: segundo jogador logado para o jogo.

Constructor Summary

Constructors

Constructor	Description
`Jogo(Jogador<T> jogadorA, Jogador<T> jogadorB, String tipo)`	Constructor para a classe Jogo.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description

`Jogador<T>`	`getJogadorA()`	Método público responsável por acessar o valor do atributo privado jogadorA.
`Jogador<T>`	`getJogadorB()`	Método público responsável por acessar o valor do atributo privado jogadorB.

Methods inherited from class java.lang.Object

`clone` , `equals` , `finalize` , `getClass` , `hashCode` , `notify` , `notifyAll` , `toString` , `wait` , `wait` , `wait`

Field Details

baralho

```
private ArrayList <T> baralho
```

ArrayList baralho: lista de cartas de um tipo de baralho específico.

jogadorA

```
private Jogador<T> jogadorA
```

Jogador jogadorA: primeiro jogador logado para o jogo.

jogadorB

```
private Jogador<T> jogadorB
```

Jogador jogadorB: segundo jogador logado para o jogo.

Constructor Details

Jogo

```
public Jogo(Jogador<T> jogadorA,
            Jogador<T> jogadorB,
            String  tipo)
     throws SQLException ,
            JogoException
```

Constructor para a classe Jogo. Inicializa os jogadores da classe (this), busca o baralho no banco de dados (tipo) e inicializa a lista de cartas da classe. Por meio de um método de Collections embaralha as cartas. Depois divide as cartas na metade para cada usuário.

Parameters:

`jogadorA` - (Jogador) primeiro jogador logado para um tipo de baralho ().

`jogadorB` - (Jogador) segundo jogador logado para um tipo de baralho ().

`tipo` - ('String') Irá definir o tipo do baralho.

Throws:

`SQLException` - Se houver problema para conectar no banco de dados.

`JogoException` - Se houver erro para inicializar o jogo (tipo não existe).

Method Details

getJogadorA

```
public Jogador<T> getJogadorA()
```

Método público responsável por acessar o valor do atributo privado jogadorA.

Returns:

(Jogador) primeiro jogador logado.

getJogadorB

```
public Jogador<T> getJogadorB()
```

Método público responsável por acessar o valor do atributo privado jogadorB.

Returns:

(Jogador) segundo jogador logado.

Module com.example.supertrunfo
Package poo.trabalhofinal.supertrunfo.classes

Class Jogador<T>

java.lang.Object
    poo.trabalhofinal.supertrunfo.classes.Jogador<T>

Type Parameters:

T - Generics que representa o tipo de jogo escolhido (baralho que está sendo utilizado).

```
public class Jogador<T>
extends Object
```

Classe Jogador

Classe responsável por aramazenar os dados de cada jogador e guardar dados como o baralho do jogador que está jogando, fazer o movimento do topo do baralho e pontuar a cada partida / rodada.

Field Summary

Fields

Modifier and Type	Field	Description
`private List <T>`	`cartas`	List cartas: tipo genérico que representa a lista de cartas de um tipo de baralho (escolhido para o jogo) que o jogador tem.
`private String`	`nome`	'String' nome: nome do jogador.
`private Integer`	`pontuacao`	Integer pontuacao: pontuação do jogador.
`private String`	`senha`	'String' se4nha: senha do jogador logado.

Constructor Summary

Constructors

Constructor	Description
`Jogador()`	Construtor vazio para a classe.
`Jogador(String nome, String senha)`	Construtor da classe que recebe como parâmetros elementos que serão atribuídos aos atributos.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
List <T>	getCartas()	Método público que permite acessar a lista de cartas disponível para o jogador (atributo privado).
String	getNome()	Método público que permite acessar o valor do atributo privado nome do Jogador.
Integer	getPontuacao()	Método público que permite acessar o valor do atributo privado pontuação.
String	getSenha()	Método público que permite acessar o valor do atributo privado senha do Jogador.
void	moveCartas(Jogador derrotado)	Método responsável por atualizar o baralho de cartas dos dois jogadores após uma partida.
void	moveTopo()	Método público que tira a carta do topo do baralho e a adiciona no final do baralho.
void	pontua(int pontos)	Método que atualiza a quantidade de pontos do jogador.
void	setCartas(List <T> cartas)	Método público que permite modificar o atributo privado baralho do jogador.
void	setNome(String nome)	Método público que permite modificar o atributo privado nome do Jogador.
void	setPontuacao(Integer pontuacao)	Método público que permite modificar o valor do campo privado pontuação do jogador.
void	setSenha(String senha)	Método público que permite modificar o atributo privado senha do Jogador.

Methods inherited from class java.lang.Object

clone , equals , finalize , getClass , hashCode , notify , notifyAll , toString , wait , wait , wait

Field Details

nome

```
private String nome
```

'String' nome: nome do jogador.

senha

```
private String  senha
```

'String' se4nha: senha do jogador logado.

pontuacao

```
private Integer  pontuacao
```

Integer pontuacao: pontuação do jogador.

cartas

```
private List <T> cartas
```

List cartas: tipo genérico que representa a lista de cartas de um tipo de baralho (escolhido para o jogo) que o jogador tem.

Constructor Details

Jogador

```
public Jogador()
```

Construtor vazio para a classe. Usado quando não há parâmetros para serem passados. Construtor *default*.

Jogador

```
public Jogador(String  nome,
               String  senha)
```

Construtor da classe que recebe como parâmetros elementos que serão atribuídos aos atributos.

Parameters:

`nome` - ('String'): nome do jogador.

`senha` - ('String'): senha do jogador.

Method Details

getNome

```
public String  getNome()
```

Método público que permite acessar o valor do atributo privado nome do Jogador.

Returns:

('String') nome do jogador.

setNome

```
public void setNome(String  nome)
```

Método público que permite modificar o atributo privado nome do Jogador.

Parameters:

`nome` - ('String'): novo nome assumido pelo jogador.

getSenha

```
public String  getSenha()
```

Método público que permite acessar o valor do atributo privado senha do Jogador.

Returns:

('String') senha do jogador.

setSenha

```
public void setSenha(String  senha)
```

Método público que permite modificar o atributo privado senha do Jogador.

Parameters:

`senha` - ('String'): nova senha assumido pelo jogador.

getPontuacao

```
public Integer  getPontuacao()
```

Método público que permite acessar o valor do atributo privado pontuação.

Returns:

(Integer) pontuação do jogador.

setPontuacao

```
public void setPontuacao(Integer  pontuacao)
```

Método público que permite modificar o valor do campo privado pontuação do jogador.

Parameters:

`pontuacao` - (Integer) nova pontuação.

getCartas

```
public List <T> getCartas()
```

Método público que permite acessar a lista de cartas disponível para o jogador (atributo privado).

Returns:

(List) baralho do jogador.

setCartas

```
public void setCartas(List <T> cartas)
```

Método público que permite modificar o atributo privado baralho do jogador.

Parameters:

`cartas` - (List) nova lista de cartas do jogador.

moveTopo

```
public void moveTopo()
```

Método público que tira a carta do topo do baralho e a adiciona no final do baralho. Usado após uma partida.

pontua

```
public void pontua(int pontos)
```

Método que atualiza a quantidade de pontos do jogador.

Parameters:

`pontos` - (int) pontos da partida (positivo se vencer e negativo se perder).

moveCartas

```
public void moveCartas(Jogador derrotado)
```

Método responsável por atualizar o baralho de cartas dos dois jogadores após uma partida.

Listas auxiliares são craiadas para armazenar ambos os baralhos (de ambos os jogadores). O topo do vencedor é removido e adicionado ao final do seu baralho. O topo do perdedor é removido e adicionado ao final do baralho do vencedor. Faz a atualização dos baralhos.

Parameters:

`derrotado` - (Jogador) jogador que perdeu a partida em relação ao jogador desta instância (this).

Module com.example.supertrunfo

Package poo.trabalhofinal.supertrunfo.classes.cartas

Class Carta

java.lang.Object
    poo.trabalhofinal.supertrunfo.classes.cartas.Carta

Direct Known Subclasses:

Gato, LinguagensProgramacao, Personagem

```
public abstract class Carta
extends Object
```

Classe Carta

Classe abstrata que contém os campos gerais compartilhados por todas as classes que herdam ela. Possui as características compartilhadas por todos os baralhos, independentemente do tipo.

Field Summary

Fields

Modifier and Type	Field	Description
private Classificacao	classificacao	Classificação classificacao: atributo privado que indica a classificação da carta.
private String	imagem	'String' url: atributo privado que indica a url da imagem da carta.
private String	nome	'String' nome: atributo privado que indica o nome da carta.
private boolean	superTrunfo	boolean superTrunfo: atributo privado que indica se a carta é ou não super trunfo.

Constructor Summary

Constructors

Constructor	Description
Carta()	Construtor vazio para a classe.
Carta(String nome, String imagem, boolean superTrunfo, Classificacao classificacao)	Construtor da superclasse abstrata, que recebe como parâmetros elementos que serão atribuídos a cada atributo.

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
Classificacao	getClassificacao()	Método público que permite acessar o valor do atributo privado *classificacao*.
String	getImagem()	Método público que permite acessar o valor do atributo privado *imagem*.
String	getNome()	Método público que permite acessar o valor do atributo privado *nome*.
boolean	isSuperTrunfo()	Método público que permite acessar o valor do atributo privado *supertrunfo*.
void	setClassificacao(String classificacao)	Método que recebe uma 'String' representando a classificação e a transforma em um tipo enum Classificacao.
void	setImagem(String imagem)	Método público que permite modificar o valor do atributo privado *imagem*.
void	setNome(String nome)	Método público que permite modificar o valor do atributo privado *nome*.
void	setSuperTrunfo(String superTrunfo)	Método público que permite modificar o valor do atributo privado *supertrunfo*.

Methods inherited from class java.lang.Object

clone , equals , finalize , getClass , hashCode , notify , notifyAll , toString , wait , wait , wait

Field Details

nome

```
private String nome
```

'String' nome: atributo privado que indica o nome da carta.

imagem

```
private String imagem
```

‘String’ url: atributo privado que indica a url da imagem da carta.

superTrunfo

```
private boolean superTrunfo
```

boolean superTrunfo: atributo privado que indica se a carta é ou não super trunfo.

classificacao

```
private Classificacao classificacao
```

Classificação classificacao: atributo privado que indica a classificação da carta.

Constructor Details

Carta

```
public Carta()
```

Construtor vazio para a classe. Usado quando não há parâmetros para serem passados. Construtor *default*.

Carta

```
public Carta(String  nome,
             String  imagem,
             boolean superTrunfo,
             Classificacao classificacao)
```

Construtor da superclasse abstrata, que recebe como parâmetros elementos que serão atribuídos a cada atributo.

Parameters:

`nome` - (‘String’) representa o nome atribuído a essa carta.

`imagem` - (‘String’) representa a url da imagem que será colocada na carta.

`superTrunfo` - (boolean) elemento que indica se a carta é um supertrunfo ou não.

`classificacao` - (Classificação) indica a classificação da carta em relação a uma ordem de classificações.

Method Details

getNome

```
public String  getNome()
```

Método público que permite acessar o valor do atributo privado *nome*.

Returns:

('String') nome associado à carta.

setNome

```
public void setNome(String  nome)
```

Método público que permite modificar o valor do atributo privado *nome*.

Parameters:

`nome` - ('String') novo nome associado à carta.

getImagem

```
public String  getImagem()
```

Método público que permite acessar o valor do atributo privado *imagem*.

Returns:

('String') url da imagem associada à carta.

setImagem

```
public void setImagem(String  imagem)
```

Método público que permite modificar o valor do atributo privado *imagem*.

Parameters:

`imagem` - ('String') nova url da imagem associada à carta.

isSuperTrunfo

```
public boolean isSuperTrunfo()
```

Método público que permite acessar o valor do atributo privado *supertrunfo*.

Returns:

(boolean) supertrunfo.

setSuperTrunfo

```
public void setSuperTrunfo(String  superTrunfo)
```

Método público que permite modificar o valor do atributo privado *supertrunfo*.

Parameters:

`superTrunfo` - (boolean) nova url da imagem associada à carta.

getClassificacao

```
public Classificacao getClassificacao()
```

Método público que permite acessar o valor do atributo privado *classificacao*.

Returns:

(Classificação) tipo enum que representa a classificação da carta.

setClassificacao

```
public void setClassificacao(String  classificacao)
```

Método que recebe uma 'String' representando a classificação e a transforma em um tipo enum Classificacao. A análise para a transformação é feita através de um switch...case.

Parameters:

`classificacao` - (‘String’) classificação da carta, para ser analisada e salvada como um tipo Classificacao.

Module com.example.supertrunfo

Package poo.trabalhofinal.supertrunfo.classes.cartas

Enum Class Classificacao

java.lang.Object
  java.lang.Enum <Classificacao>
    poo.trabalhofinal.supertrunfo.classes.cartas.Classificacao

All Implemented Interfaces:

`Serializable` , `Comparable` <`Classificacao`>, `Constable`

```
public enum Classificacao
extends Enum <Classificacao>
```

Enum Classificação

Classe que representa a classificação de uma carta de forma ordenada. D5 e o mais fraco e A1 é o mais forte. Classificação crescente.

Nested Class Summary

Nested classes/interfaces inherited from class java.lang.Enum

`Enum.EnumDesc` <E extends `Enum` <E >>

Enum Constant Summary

Enum Constants

Enum Constant	Description
A1	
A2	
A3	
A4	
A5	
B1	
B2	
B3	
B4	
B5	
C1	

C2

C3

C4

C5

D1

D2

D3

D4

D5

Constructor Summary

Constructors

Modifier	Constructor	Description
private	Classificacao()	

Method Summary

All Methods | Static Methods | Concrete Methods

Modifier and Type	Method	Description
static Classificacao	valueOf(String name)	Returns the enum constant of this class with the specified name.
static Classificacao[]	values()	Returns an array containing the constants of this enum class, in the order they are declared.

Methods inherited from class java.lang.Enum

clone , compareTo , describeConstable , equals , finalize , getDeclaringClass , hashCode , name , ordinal , toString , valueOf

Methods inherited from class java.lang.Object

getClass , notify , notifyAll , wait , wait , wait

Enum Constant Details

D5

```
public static final Classificacao D5
```

D4

```
public static final Classificacao D4
```

D3

```
public static final Classificacao D3
```

D2

```
public static final Classificacao D2
```

D1

```
public static final Classificacao D1
```

C5

```
public static final Classificacao C5
```

C4

```
public static final Classificacao C4
```

C3

```
public static final Classificacao C3
```

C2

```
public static final Classificacao C2
```

C1

```
public static final Classificacao C1
```

B5

```
public static final Classificacao B5
```

B4

```
public static final Classificacao B4
```

B3

```
public static final Classificacao B3
```

B2

```
public static final Classificacao B2
```

B1

```
public static final Classificacao B1
```

A5

```
public static final Classificacao A5
```

A4

```
public static final Classificacao A4
```

A3

```
public static final Classificacao A3
```

A2

```
public static final Classificacao A2
```

A1

```
public static final Classificacao A1
```

Constructor Details

Classificacao

```
private Classificacao()
```

Method Details

values

```
public static Classificacao[] values()
```

Returns an array containing the constants of this enum class, in the order they are declared.

Returns:

an array containing the constants of this enum class, in the order they are declared

valueOf

```
public static Classificacao valueOf(String  name)
```

Returns the enum constant of this class with the specified name. The string must match *exactly* an identifier used to declare an enum constant in this class. (Extraneous whitespace characters are not permitted.)

Parameters:

`name` - the name of the enum constant to be returned.

Returns:

the enum constant with the specified name

Throws:

`IllegalArgumentException` - if this enum class has no constant with the specified name

`NullPointerException` - if the argument is null

Module com.example.supertrunfo

Package poo.trabalhofinal.supertrunfo.classes.cartas

Class Gato

java.lang.Object
    poo.trabalhofinal.supertrunfo.classes.cartas.Carta
        poo.trabalhofinal.supertrunfo.classes.cartas.Gato

```
public class Gato
extends Carta
```

Classe Gato

Subclasse que herda atributos e métodos da superclasse abstrata Carta. É um tipo específico de carta do jogo. Possui atributos e métodos próprios e inerentes a cartas do baralho de Gato.

Field Summary

Fields

Modifier and Type	Field	Description
`private Integer`	`agilidade`	'Integer' agilidade: atributo privado associado à carta do tipo Gato, tendo valor de 0 a 100.
`private Integer`	`agressividade`	'Integer' agressividade: atributo privado associado à carta do tipo Gato, tendo valor de 0 a 100.
`private Integer`	`fofura`	'Integer' fofura: atributo privado associado à carta do tipo Gato, tendo valor de 0 a 100.
`private Double`	`peso`	'Double' peso: atributo privado associado à carta do tipo Gato, tendo valor em kg.
`private Integer`	`tempoDeVida`	'Integer' tempoDeVida: atributo privado associado à carta do tipo Gato, representando a idade em meses.

Constructor Summary

Constructors

Constructor	Description
`Gato()`	Construtor vazio para a classe.
`Gato(String nome, String imagem, boolean superTrunfo, Classificacao classificacao, Integer agilidade, Integer fofura, Integer`	Construtor que recebe elementos a serem associados aos atributos da classe (e da superclasse) como parâmetro.

```
 tempoDeVida, Integer  agressividade,
Double  peso)
```

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
int	comparaAgilidade (Gato topoOponente)	Método cuja função é fazer a comparação entre a agilidade da carta (this) atual com a outra carta adversária (ambas no topo do baralho).
int	comparaAgressividade (Gato topoOponente)	Método cuja função é fazer a comparação entre a agressividade da carta (this) atual com a outra carta adversária (ambas no topo do baralho).
int	comparaFofura(Gato topoOponente)	Método cuja função é fazer a comparação entre a fofura da carta (this) atual com a outra carta adversária (ambas no topo do baralho).
int	comparaPeso(Gato topoOponente)	Método cuja função é fazer a comparação entre o peso da carta (this) atual com a outra carta adversária (ambas no topo do baralho).
int	comparaVida(Gato topoOponente)	Método cuja função é fazer a comparação entre a idade da carta (this) atual com a outra carta adversária (ambas no topo do baralho).
Integer	getAgilidade()	Método público que permite acessar o valor do atributo privado *agilidade*.
Integer	getAgressividade()	Método público que permite acessar o valor do atributo privado *agressividade*.
Integer	getFofura()	Método público que permite acessar o valor do atributo privado *fofura*.
Double	getPeso()	Método público que permite acessar o valor do atributo privado *peso*.
Integer	getTempoDeVida()	Método público que permite acessar o valor do atributo privado

		tempoDeVida.
void	**setAgilidade**(**Integer** agilidade)	Método público que permite modificar o valor do atributo privado *agilidade*.
void	**setAgressividade**(**Integer** agressividade)	Método público que permite modificar o valor do atributo privado *agressividade*.
void	**setFofura**(**Integer** fofura)	Método público que permite modificar o valor do atributo privado *fofura*.
void	**setPeso**(**Double** peso)	Método público que permite modificar o valor do atributo privado *peso*.
void	**setTempoDeVida**(**Integer** tempoDeVida)	Método público que permite modificar o valor do atributo privado *tempoDeVida*.
String	**toString**()	Método que devolve o tipo da carta (no caso é tipo Gato).

Methods inherited from class poo.trabalhofinal.supertrunfo.classes.cartas.Carta

`getClassificacao`, `getImagem`, `getNome`, `isSuperTrunfo`, `setClassificacao`, `setImagem`, `setNome`, `setSuperTrunfo`

Methods inherited from class java.lang.Object

`clone` , `equals` , `finalize` , `getClass` , `hashCode` , `notify` , `notifyAll` , `wait` , `wait` , `wait`

Field Details

agilidade

```
private Integer  agilidade
```

'Integer' agilidade: atributo privado associado à carta do tipo Gato, tendo valor de 0 a 100.

fofura

```
private Integer  fofura
```

'Integer' fofura: atributo privado associado à carta do tipo Gato, tendo valor de 0 a 100.

tempoDeVida

```
private Integer  tempoDeVida
```

'Integer' tempoDeVida: atributo privado associado à carta do tipo Gato, representando a idade em meses.

agressividade

```
private Integer  agressividade
```

'Integer' agressividade: atributo privado associado à carta do tipo Gato, tendo valor de 0 a 100.

peso

```
private Double  peso
```

'Double' peso: atributo privado associado à carta do tipo Gato, tendo valor em kg.

Constructor Details

Gato

```
public Gato()
```

Construtor vazio para a classe. Usado quando não há parâmetros para serem passados. Construtor *default*.

Gato

```
public Gato(String  nome,
            String  imagem,
            boolean superTrunfo,
            Classificacao classificacao,
            Integer  agilidade,
            Integer  fofura,
            Integer  tempoDeVida,
            Integer  agressividade,
            Double  peso)
```

Construtor que recebe elementos a serem associados aos atributos da classe (e da superclasse) como parâmetro.

Parameters:

`nome` - (‘String’) representa o nome atribuído a essa carta.

`imagem` - (‘String’) representa a url da imagem que será colocada na carta.

`superTrunfo` - (boolean) elemento que indica se a carta é um supertrunfo ou não.

`classificacao` - (Classificação) indica a classificação da carta em relação a uma ordem de classificações.

`agilidade` - ('Integer') indica o valor de agilidade associado à carta do baralho.

`fofura` - ('Integer') indica o valor de fofura associado à carta do baralho.

`tempoDeVida` - ('Integer') indica o valor da idade (meses) associado à carta do baralho.

`agressividade` - ('Integer') indica o valor de agressividade associado à carta do baralho.

`peso` - ('Double') indica o valor do peso (kg) associado à carta do baralho.

Method Details

getAgilidade

```
public Integer  getAgilidade()
```

Método público que permite acessar o valor do atributo privado *agilidade*.

Returns:

('Integer') agilidade do gato.

setAgilidade

```
public void setAgilidade(Integer  agilidade)
```

Método público que permite modificar o valor do atributo privado *agilidade*.

Parameters:

`agilidade` - ('Integer') novo valor de agilidade associado à carta.

getFofura

```
public Integer  getFofura()
```

Método público que permite acessar o valor do atributo privado *fofura*.

Returns:

('Integer') fofura do gato.

setFofura

```
public void setFofura(Integer  fofura)
```

Método público que permite modificar o valor do atributo privado *fofura*.

Parameters:

`fofura` - ('Integer') novo valor de fofura associado à carta.

getTempoDeVida

```
public Integer  getTempoDeVida()
```

Método público que permite acessar o valor do atributo privado *tempoDeVida*.

Returns:

('Integer') idade do gato (em meses).

setTempoDeVida

```
public void setTempoDeVida(Integer  tempoDeVida)
```

Método público que permite modificar o valor do atributo privado *tempoDeVida*.

Parameters:

`tempoDeVida` - ('Integer') nova idade associada à carta (em meses).

getAgressividade

```
public Integer  getAgressividade()
```

Método público que permite acessar o valor do atributo privado *agressividade*.

Returns:

('Integer') agressividade do gato.

setAgressividade

```
public void setAgressividade(Integer  agressividade)
```

Método público que permite modificar o valor do atributo privado *agressividade*.

Parameters:

`agressividade` - ('Integer') novo valor de agressividade associado à carta.

getPeso

```
public Double  getPeso()
```

Método público que permite acessar o valor do atributo privado *peso*.

Returns:

('Double') peso do gato (em kg).

setPeso

```
public void setPeso(Double  peso)
```

Método público que permite modificar o valor do atributo privado *peso*.

Parameters:

`peso` - ('Integer') novo peso associado à carta (em kg).

comparaAgilidade

```
public int comparaAgilidade(Gato topoOponente)
```

Método cuja função é fazer a comparação entre a agilidade da carta (this) atual com a outra carta adversária (ambas no topo do baralho). A carta vencedora será a que possui o maior valor no atributo agilidade.

Parameters:

`topoOponente` - (Gato) carta do tipo gato com a qual faremos a comparação.

Returns:

(int) 0 se forem o mesmo valor, 1 se esta (this) for maior em agilidade e -1 se a carta oponente for maior que esta (this).

comparaFofura

```
public int comparaFofura(Gato topoOponente)
```

Método cuja função é fazer a comparação entre a fofura da carta (this) atual com a outra carta adversária (ambas no topo do baralho). A carta vencedora será a que possui o maior valor no atributo fofura.

Parameters:

`topoOponente` - (Gato) carta do tipo gato com a qual faremos a comparação.

Returns:

(int) 0 se forem o mesmo valor, 1 se esta (this) for maior em fofura e -1 se a carta oponente for maior que esta (this).

comparaVida

```
public int comparaVida(Gato topoOponente)
```

Método cuja função é fazer a comparação entre a idade da carta (this) atual com a outra carta adversária (ambas no topo do baralho). A carta vencedora será a que possui o maior valor no atributo idade.

Parameters:

`topoOponente` - (Gato) carta do tipo gato com a qual faremos a comparação.

Returns:

(int) 0 se forem o mesmo valor, 1 se esta (this) for maior em idade e -1 se a carta oponente for maior que esta (this).

comparaAgressividade

```
public int comparaAgressividade(Gato topoOponente)
```

Método cuja função é fazer a comparação entre a agressividade da carta (this) atual com a outra carta adversária (ambas no topo do baralho). A carta vencedora será a que possui o maior valor no atributo agressividade.

Parameters:

`topoOponente` - (Gato) carta do tipo gato com a qual faremos a comparação.

Returns:

(int) 0 se forem o mesmo valor, 1 se esta (this) for maior em agressividade e -1 se a carta oponente for maior que esta (this).

comparaPeso

```
public int comparaPeso(Gato topoOponente)
```

Método cuja função é fazer a comparação entre o peso da carta (this) atual com a outra carta adversária (ambas no topo do baralho). A carta vencedora será a que possui o maior valor no atributo peso.

Parameters:

`topoOponente` - (Gato) carta do tipo gato com a qual faremos a comparação.

Returns:

(int) 0 se forem o mesmo valor, 1 se esta (this) for maior em peso e -1 se a carta oponente for maior que esta (this).

toString

```
public String  toString()
```

Método que devolve o tipo da carta (no caso é tipo Gato).

Overrides:

`toString` in class `Object`

Returns:

('String') tipo da carta.

Module com.example.supertrunfo

Package poo.trabalhofinal.supertrunfo.classes.cartas

Class LinguagensProgramacao

java.lang.Object
poo.trabalhofinal.supertrunfo.classes.cartas.Carta
poo.trabalhofinal.supertrunfo.classes.cartas.LinguagensProgramacao

---

```
public class LinguagensProgramacao
extends Carta
```

Classe LinguagensProgramacao

Subclasse que herda atributos e métodos da superclasse abstrata Carta. É um tipo específico de carta do jogo. Possui atributos e métodos próprios e inerentes a cartas do baralho de LinguagensProgramacao.

Field Summary

Fields

Modifier and Type	Field	Description
`private Integer`	`confiabilidade`	'Integer' confiabilidade: atributo privado associado à carta do tipo LinguagensProgramacao, tendo um valor de 0 a 100.
`private Integer`	`custo`	'Integer' custo: atributo privado associado à carta do tipo LinguagensProgramacao, tendo um valor de 0 a 100.
`private Integer`	`escritabilidade`	'Integer' escritabilidade: atributo privado associado à carta do tipo LinguagensProgramacao, tendo um valor de 0 a 100.
`private Integer`	`legibilidade`	'Integer' legibilidade: atributo privado associado à carta do tipo LinguagensProgramacao, tendo um valor de 0 a 100.
`private Double`	`salarioSenior`	'Double' salarioSenior: atributo privado associado à carta do tipo LinguagensProgramacao, sendo um valor em reais.

Constructor Summary

Constructors

Constructor	Description
`LinguagensProgramacao()`	Construtor vazio para a classe.

`LinguagensProgramacao(String nome, String imagem, boolean superTrunfo, Classificacao classificacao, Integer escritabilidade, Integer legibilidade, Integer confiabilidade, Integer custo, Double salarioSenior)`	Construtor que recebe elementos a serem associados aos atributos da classe (e da superclasse) como parâmetro.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
int	`comparaConfiabilidade(LinguagensProgramacao oponente)`	Método cuja função é fazer a comparação entre a confiabilidade da carta (this) atual com a outra carta adversária.
int	`comparaCusto(LinguagensProgramacao oponente)`	Método cuja função é fazer a comparação entre o custo da carta (this) atual com a outra carta adversária.
int	`comparaEscritabilidade(LinguagensProgramacao oponente)`	Método cuja função é fazer a comparação entre a escritabilidade da carta (this) atual com a outra carta adversária.
int	`comparaLegibilidade(LinguagensProgramacao oponente)`	Método cuja função é fazer a comparação entre a legibilidade da carta (this) atual com a outra carta adversária.
int	`comparaSalario(LinguagensProgramacao oponente)`	Método cuja função é fazer a comparação entre o salarioSenior da carta (this) atual com a outra carta adversária.
`Integer`	`getConfiabilidade()`	Método público que permite acessar o valor do atributo privado *confiabilidade*.
`Integer`	`getCusto()`	Método público que permite acessar o valor do atributo privado *custo*.
`Integer`	`getEscritabilidade()`	Método público que permite acessar o valor do atributo privado *escritabilidade*.
`Integer`	`getLegibilidade()`	Método público que permite acessar o valor do atributo privado *legibilidade*.

Double	getSalarioSenior()	Método público que permite acessar o valor do atributo privado *salarioSenior*.
void	setConfiabilidade(Integer confiabilidade)	Método público que permite modificar o valor do atributo privado *confiabilidade*.
void	setCusto(Integer custo)	Método público que permite modificar o valor do atributo privado *custo*.
void	setEscritabilidade(Integer escritabilidade)	Método público que permite modificar o valor do atributo privado *escritabilidade*.
void	setLegibilidade(Integer legibilidade)	Método público que permite modificar o valor do atributo privado *legibilidade*.
void	setSalarioSenior(Double salarioSenior)	Método público que permite modificar o valor do atributo privado *salarioSenior*.
String	toString()	Método que devolve o tipo da carta (no caso é tipo LinguagensProgramacao).

Methods inherited from class poo.trabalhofinal.supertrunfo.classes.cartas.Carta

getClassificacao, getImagem, getNome, isSuperTrunfo, setClassificacao, setImagem, setNome, setSuperTrunfo

Methods inherited from class java.lang.Object

clone , equals , finalize , getClass , hashCode , notify , notifyAll , wait , wait , wait

Field Details

escritabilidade

```
private Integer  escritabilidade
```

'Integer' escritabilidade: atributo privado associado à carta do tipo LinguagensProgramacao, tendo um valor de 0 a 100. Representa a facilidade de escrever um código nesta linguagem.

legibilidade

```
private Integer  legibilidade
```

'Integer' legibilidade: atributo privado associado à carta do tipo LinguagensProgramacao, tendo um valor de 0 a 100. Representa a facilidade de ler e entender um código nesta linguagem.

confiabilidade

```
private Integer  confiabilidade
```

'Integer' confiabilidade: atributo privado associado à carta do tipo LinguagensProgramacao, tendo um valor de 0 a 100. Representa o nível de confiabilidade de um programa desta linguagem.

custo

```
private Integer  custo
```

'Integer' custo: atributo privado associado à carta do tipo LinguagensProgramacao, tendo um valor de 0 a 100. Representa o custo computacional de um programa desta linguagem.

salarioSenior

```
private Double  salarioSenior
```

'Double' salarioSenior: atributo privado associado à carta do tipo LinguagensProgramacao, sendo um valor em reais. Representa o salário que um programador sênior desta linguagem pode receber.

Constructor Details

LinguagensProgramacao

```
public LinguagensProgramacao()
```

Construtor vazio para a classe. Usado quando não há parâmetros para serem passados. Construtor *default*.

LinguagensProgramacao

```
public LinguagensProgramacao(String  nome,
                             String  imagem,
                             boolean superTrunfo,
                             Classificacao classificacao,
                             Integer  escritabilidade,
                             Integer  legibilidade,
                             Integer  confiabilidade,
                             Integer  custo,
                             Double  salarioSenior)
```

Construtor que recebe elementos a serem associados aos atributos da classe (e da superclasse) como parâmetro.

Parameters:

`nome` - (‘String’) representa o nome atribuído a essa carta.

`imagem` - (‘String’) representa a url da imagem que será colocada na carta.

`superTrunfo` - (boolean) elemento que indica se a carta é um supertrunfo ou não.

`classificacao` - (Classificação) indica a classificação da carta em relação a uma ordem de classificações.

`escritabilidade` - ('Integer') indica o valor de escritabilidade associado à carta do baralho.

`legibilidade` - ('Integer') indica o valor de legibilidade associado à carta do baralho.

`confiabilidade` - ('Integer') indica o valor de confiabilidade associado à carta do baralho.

`custo` - ('Integer') indica o valor de custo associado à carta do baralho.

`salarioSenior` - ('Double') indica o valor do salário de um prgramador sênior associado à carta do baralho.

Method Details

getEscritabilidade

```
public Integer  getEscritabilidade()
```

Método público que permite acessar o valor do atributo privado *escritabilidade*.

Returns:

('Integer') escritabilidade da linguagem de programação.

setEscritabilidade

```
public void setEscritabilidade(Integer  escritabilidade)
```

Método público que permite modificar o valor do atributo privado *escritabilidade*.

Parameters:

`escritabilidade` - ('Integer') novo valor de escritabilidade associado à carta.

getLegibilidade

```
public Integer  getLegibilidade()
```

Método público que permite acessar o valor do atributo privado *legibilidade*.

Returns:

('Integer') legibilidade da linguagem de programação.

setLegibilidade

```
public void setLegibilidade(Integer  legibilidade)
```

Método público que permite modificar o valor do atributo privado *legibilidade*.

Parameters:

`legibilidade` - ('Integer') novo valor de legibilidade associado à carta.

getConfiabilidade

```
public Integer  getConfiabilidade()
```

Método público que permite acessar o valor do atributo privado *confiabilidade*.

Returns:

('Integer') confiabilidade da linguagem de programação.

setConfiabilidade

```
public void setConfiabilidade(Integer  confiabilidade)
```

Método público que permite modificar o valor do atributo privado *confiabilidade*.

Parameters:

`confiabilidade` - ('Integer') novo valor de confiabilidade associado à carta.

getCusto

```
public Integer  getCusto()
```

Método público que permite acessar o valor do atributo privado *custo*.

Returns:

('Integer') custo da linguagem de programação.

setCusto

```
public void setCusto(Integer  custo)
```

Método público que permite modificar o valor do atributo privado *custo*.

Parameters:

`custo` - ('Integer') novo valor de custo associado à carta.

getSalarioSenior

```
public Double  getSalarioSenior()
```

Método público que permite acessar o valor do atributo privado *salarioSenior*.

Returns:

('Double') salarioSenior da linguagem de programação (em reais).

setSalarioSenior

```
public void setSalarioSenior(Double  salarioSenior)
```

Método público que permite modificar o valor do atributo privado *salarioSenior*.

Parameters:

`salarioSenior` - ('Double') novo valor de salarioSenior associado à carta (em reais).

comparaEscritabilidade

```
public int comparaEscritabilidade(LinguagensProgramacao oponente)
```

Método cuja função é fazer a comparação entre a escritabilidade da carta (this) atual com a outra carta adversária. A carta vencedora será a que possui o maior valor no atributo escritabilidade.

Parameters:

`oponente` - (LinguagensProgramacao) carta do tipo gato com a qual faremos a comparação.

Returns:

(int) 0 se forem o mesmo valor, 1 se esta (this) for maior em escritabilidade e -1 se a carta oponente for maior que esta (this).

comparaLegibilidade

```
public int comparaLegibilidade(LinguagensProgramacao oponente)
```

Método cuja função é fazer a comparação entre a legibilidade da carta (this) atual com a outra carta adversária. A carta vencedora será a que possui o maior valor no atributo legibilidade.

Parameters:

`oponente` - (LinguagensProgramacao) carta do tipo gato com a qual faremos a comparação.

Returns:

(int) 0 se forem o mesmo valor, 1 se esta (this) for maior em legibilidade e -1 se a carta oponente for maior que esta (this).

comparaConfiabilidade

```
public int comparaConfiabilidade(LinguagensProgramacao oponente)
```

Método cuja função é fazer a comparação entre a confiabilidade da carta (this) atual com a outra carta adversária. A carta vencedora será a que possui o maior valor no atributo confiabilidade.

Parameters:

`oponente` - (LinguagensProgramacao) carta do tipo gato com a qual faremos a comparação.

Returns:

(int) 0 se forem o mesmo valor, 1 se esta (this) for maior em confiabilidade e -1 se a carta oponente for maior que esta (this).

comparaCusto

```
public int comparaCusto(LinguagensProgramacao oponente)
```

Método cuja função é fazer a comparação entre o custo da carta (this) atual com a outra carta adversária. A carta vencedora será a que possui o maior valor no atributo custo.

Parameters:

`oponente` - (LinguagensProgramacao) carta do tipo gato com a qual faremos a comparação.

Returns:

(int) 0 se forem o mesmo valor, 1 se esta (this) for maior em custo e -1 se a carta oponente for maior que esta (this).

comparaSalario

```
public int comparaSalario(LinguagensProgramacao oponente)
```

Método cuja função é fazer a comparação entre o salarioSenior da carta (this) atual com a outra carta adversária. A carta vencedora será a que possui o maior valor no atributo salarioSenior.

Parameters:

`oponente` - (LinguagensProgramacao) carta do tipo gato com a qual faremos a comparação.

Returns:

(int) 0 se forem o mesmo valor, 1 se esta (this) for maior em salarioSenior e -1 se a carta oponente for maior que esta (this).

toString

```
public String  toString()
```

Método que devolve o tipo da carta (no caso é tipo LinguagensProgramacao).

Overrides:

`toString` in class `Object`

Returns:

('String') tipo da carta.

Module com.example.supertrunfo

Package poo.trabalhofinal.supertrunfo.classes.cartas

Class Personagem

java.lang.Object
    poo.trabalhofinal.supertrunfo.classes.cartas.Carta
        poo.trabalhofinal.supertrunfo.classes.cartas.Personagem

---

```
public class Personagem
extends Carta
```

Classe Personagem

Subclasse que herda atributos e métodos da superclasse abstrata Carta. É um tipo específico de carta do jogo. Possui atributos e métodos próprios e inerentes a cartas do baralho de Personagem.

Field Summary

Fields

Modifier and Type	Field	Description
private Double	altura	'Double' altura: atributo privado associado à carta do tipo Personagem, tendo valor em metros.
private Integer	coragem	'Integer' coragem: atributo privado associado à carta do tipo Personagem, tendo valor de 0 a 100.
private Integer	forca	'Integer' força: atributo privado associado à carta do tipo Personagem, tendo valor de 0 a 100.
private Integer	inteligencia	'Integer' inteligencia: atributo privado associado à carta do tipo Personagem, tendo valor de 0 a 100.
private Integer	primeiraAparicao	'Integer' primeiraAparicao: atributo privado associado à carta do tipo Personagem, representando o ano em que o personagem primeiro apareceu.

Constructor Summary

Constructors

Constructor	Description
Personagem()	Construtor vazio para a classe.
Personagem(String nome, String imagem, boolean superTrunfo, Classificacao classificacao, Integer inteligencia, Integer forca, Integer	Construtor que recebe elementos a serem associados aos atributos da classe (e da superclasse) como parâmetro.

```
 coragem, Integer  primeiraAparicao,
Double  altura)
```

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
int	comparaAltura (Personagem topoOponente)	Método cuja função é fazer a comparação entre a altura da carta (this) atual com a outra carta adversária (ambas no topo do baralho).
int	comparaAparicao (Personagem topoOponente)	Método cuja função é fazer a comparação entre a primeira aparição da carta (this) atual com a outra carta adversária (ambas no topo do baralho).
int	comparaCoragem (Personagem topoOponente)	Método cuja função é fazer a comparação entre a coragem da carta (this) atual com a outra carta adversária (ambas no topo do baralho).
int	comparaForca (Personagem topoOponente)	Método cuja função é fazer a comparação entre a força da carta (this) atual com a outra carta adversária (ambas no topo do baralho).
int	comparaInteligencia (Personagem topoOponente)	Método cuja função é fazer a comparação entre a inteligência da carta (this) atual com a outra carta adversária (ambas no topo do baralho).
Double	getAltura()	Método público que permite acessar o valor do atributo privado *altura*.
Integer	getCoragem()	Método público que permite acessar o valor do atributo privado *coragem*.
Integer	getForca()	Método público que permite acessar o valor do atributo privado *forca*.
Integer	getInteligencia()	Método público que permite acessar o valor do atributo privado *inteligencia*.
Integer	getPrimeiraAparicao()	Método público que permite acessar o valor do atributo privado *primeiraAparicao*.

void	**setAltura**(**Double** altura)	Método público que permite modificar o valor do atributo privado *altura*.
void	**setCoragem**(**Integer** coragem)	Método público que permite modificar o valor do atributo privado *coragem*.
void	**setForca**(**Integer** forca)	Método público que permite modificar o valor do atributo privado *forca*.
void	**setInteligencia**(**Integer** inteligencia)	Método público que permite modificar o valor do atributo privado *inteligencia*.
void	**setPrimeiraAparicao**(**Integer** primeiraAparicao)	Método público que permite modificar o valor do atributo privado *primeiraAparicao*.
String	**toString**()	Método que devolve o tipo da carta (no caso é tipo Personagem).

Methods inherited from class poo.trabalhofinal.supertrunfo.classes.cartas.Carta

getClassificacao, getImagem, getNome, isSuperTrunfo, setClassificacao, setImagem, setNome, setSuperTrunfo

Methods inherited from class java.lang.Object

clone , equals , finalize , getClass , hashCode , notify , notifyAll , wait , wait , wait

Field Details

inteligencia

```
private Integer  inteligencia
```

'Integer' inteligencia: atributo privado associado à carta do tipo Personagem, tendo valor de 0 a 100.

forca

```
private Integer  forca
```

'Integer' força: atributo privado associado à carta do tipo Personagem, tendo valor de 0 a 100.

coragem

```
private Integer  coragem
```

'Integer' coragem: atributo privado associado à carta do tipo Personagem, tendo valor de 0 a 100.

primeiraAparicao

```
private Integer  primeiraAparicao
```

'Integer' primeiraAparicao: atributo privado associado à carta do tipo Personagem, representando o ano em que o personagem primeiro apareceu.

altura

```
private Double  altura
```

'Double' altura: atributo privado associado à carta do tipo Personagem, tendo valor em metros.

Constructor Details

Personagem

```
public Personagem()
```

Construtor vazio para a classe. Usado quando não há parâmetros para serem passados. Construtor *default*.

Personagem

```
public Personagem(String  nome,
                  String  imagem,
                  boolean superTrunfo,
                  Classificacao classificacao,
                  Integer  inteligencia,
                  Integer  forca,
                  Integer  coragem,
                  Integer  primeiraAparicao,
                  Double  altura)
```

Construtor que recebe elementos a serem associados aos atributos da classe (e da superclasse) como parâmetro.

Parameters:

`nome` - (‘String’) representa o nome atribuído a essa carta.

`imagem` - (‘String’) representa a url da imagem que será colocada na carta.

`superTrunfo` - (boolean) elemento que indica se a carta é um supertrunfo ou não.

`classificacao` - (Classificação) indica a classificação da carta em relação a uma ordem de classificações.

`inteligencia` - ('Integer') indica o valor de inteligência associado à carta do baralho.

`forca` - ('Integer') indica o valor de força associado à carta do baralho.

`coragem` - ('Integer') indica o valor da coragem associado à carta do baralho.

`primeiraAparicao` - ('Integer') indica o ano da primeira aparição associado à carta do baralho.

`altura` - ('Double') indica o valor da altura (metros) associado à carta do baralho.

Method Details

getInteligencia

```
public Integer  getInteligencia()
```

Método público que permite acessar o valor do atributo privado *inteligencia*.

Returns:

('Integer') inteligencia do personagem.

setInteligencia

```
public void setInteligencia(Integer  inteligencia)
```

Método público que permite modificar o valor do atributo privado *inteligencia*.

Parameters:

`inteligencia` - ('Integer') novo valor de inteligência associado à carta.

getForca

```
public Integer  getForca()
```

Método público que permite acessar o valor do atributo privado *forca*.

Returns:

('Integer') força do personagem.

setForca

```
public void setForca(Integer  forca)
```

Método público que permite modificar o valor do atributo privado *forca*.

Parameters:

`forca` - ('Integer') novo valor de força associado à carta.

getCoragem

```
public Integer  getCoragem()
```

Método público que permite acessar o valor do atributo privado *coragem*.

Returns:

('Integer') coragem do personagem.

setCoragem

```
public void setCoragem(Integer  coragem)
```

Método público que permite modificar o valor do atributo privado *coragem*.

Parameters:

`coragem` - ('Integer') novo valor de coragem associado à carta.

getPrimeiraAparicao

```
public Integer  getPrimeiraAparicao()
```

Método público que permite acessar o valor do atributo privado *primeiraAparicao*.

Returns:

('Integer') primeira aparição do personagem.

setPrimeiraAparicao

```
public void setPrimeiraAparicao(Integer  primeiraAparicao)
```

Método público que permite modificar o valor do atributo privado *primeiraAparicao*.

Parameters:

`primeiraAparicao` - ('Integer') novo valor de primeiraAparicao associado à carta.

getAltura

```
public Double  getAltura()
```

Método público que permite acessar o valor do atributo privado *altura*.

Returns:

('Double') altura (em metros) do personagem.

setAltura

```
public void setAltura(Double  altura)
```

Método público que permite modificar o valor do atributo privado *altura*.

Parameters:

`altura` - ('Double') novo valor de altura (em metros) associado à carta.

comparaInteligencia

```
public int comparaInteligencia(Personagem topoOponente)
```

Método cuja função é fazer a comparação entre a inteligência da carta (this) atual com a outra carta adversária (ambas no topo do baralho). A carta vencedora será a que possui o maior valor no atributo inteligência.

Parameters:

`topoOponente` - (Personagem) carta do tipo gato com a qual faremos a comparação.

Returns:

(int) 0 se forem o mesmo valor, 1 se esta (this) for maior em inteligência e -1 se a carta oponente for maior que esta (this).

comparaForca

```
public int comparaForca(Personagem topoOponente)
```

Método cuja função é fazer a comparação entre a força da carta (this) atual com a outra carta adversária (ambas no topo do baralho). A carta vencedora será a que possui o maior valor no atributo força.

Parameters:

`topoOponente` - (Personagem) carta do tipo gato com a qual faremos a comparação.

Returns:

(int) 0 se forem o mesmo valor, 1 se esta (this) for maior em força e -1 se a carta oponente for maior que esta (this).

comparaCoragem

```
public int comparaCoragem(Personagem topoOponente)
```

Método cuja função é fazer a comparação entre a coragem da carta (this) atual com a outra carta adversária (ambas no topo do baralho). A carta vencedora será a que possui o maior valor no atributo coragem.

Parameters:

`topoOponente` - (Personagem) carta do tipo gato com a qual faremos a comparação.

Returns:

(int) 0 se forem o mesmo valor, 1 se esta (this) for maior em coragem e -1 se a carta oponente for maior que esta (this).

comparaAparicao

```
public int comparaAparicao(Personagem topoOponente)
```

Método cuja função é fazer a comparação entre a primeira aparição da carta (this) atual com a outra carta adversária (ambas no topo do baralho). A carta vencedora será a que possui o menor valor no

atributo primeira aparição (apareceu primeiro).

Parameters:

`topoOponente` - (Personagem) carta do tipo gato com a qual faremos a comparação.

Returns:

(int) 0 se forem o mesmo valor, -1 se esta (this) for maior em primeira aparição (perde) e 1 se a carta oponente for menor (ganha) que esta (this).

comparaAltura

```
public int comparaAltura(Personagem topoOponente)
```

Método cuja função é fazer a comparação entre a altura da carta (this) atual com a outra carta adversária (ambas no topo do baralho). A carta vencedora será a que possui o maior valor no atributo altura.

Parameters:

`topoOponente` - (Personagem) carta do tipo gato com a qual faremos a comparação.

Returns:

(int) 0 se forem o mesmo valor, 1 se esta (this) for maior em altura e -1 se a carta oponente for maior que esta (this).

toString

```
public String  toString()
```

Método que devolve o tipo da carta (no caso é tipo Personagem).

Overrides:

`toString` in class `Object`

Returns:

('String') tipo da carta.

Module com.example.supertrunfo
Package poo.trabalhofinal.supertrunfo.classes.interfaces

Interface CartasRepository<T>

Type Parameters:

T - tipo genérico que vai ser preenchido por um tipo de carta.

All Known Implementing Classes:

CartasRepositoryImpl

```
public interface CartasRepository<T>
```

Interface CartasRepository

Interface que utiliza Generics e define o contrato (assinaturas dos métodos) que deve ser cumprido na classe *CartasRepositoryImpl*. Utiliza Generics porque existem 3 tipos de cartas (tipos dos baralhos) que podem ser utilizados.

Method Summary

All Methods Instance Methods Abstract Methods

Modifier and Type	Method	Description
ArrayList <T>	buscaCartas(String jogo)	Assinatura do método responsável por buscar as cartas para iniciar o jogo (busca o baralho).
ArrayList <Carta>	buscaTodasCartas()	Assinatura do método que pega todas as cartas do jogo.
void	insereNovaCarta(Gato novaCarta)	Assinatura do método que faz a inserção de uma nova carta do tipo Gato.
void	insereNovaCarta (LinguagensProgramacao novaCarta	Assinatura do método que faz a inserção de uma nova carta do tipo LinguagensProgramacao.
void	insereNovaCarta (Personagem novaCarta)	Assinatura do método que faz a inserção de uma nova carta do tipo Personagem.

Method Details

buscaCartas

```
ArrayList <T> buscaCartas(String  jogo)
                  throws SQLException ,
                         JogoException
```

Assinatura do método responsável por buscar as cartas para iniciar o jogo (busca o baralho).

Parameters:

`jogo` - ('String') que representa o tipo de baralho que está sendo jogado.

Returns:

('ArrayList') baralho de cartas que está sendo utilizado no jogo.

Throws:

`SQLException` - Se der problema com a leitura do banco de dados.

`JogoException` - Se a 'String' jogo fornecida for inválida (não achar um jogo/baralho daquele tipo).

insereNovaCarta

```
void insereNovaCarta(Personagem novaCarta)
              throws SQLException
```

Assinatura do método que faz a inserção de uma nova carta do tipo Personagem.

Parameters:

`novaCarta` - (Personagem) nova carta a ser inserida no baralho.

Throws:

`SQLException` - Se der problema ao inicializar o banco de dados.

insereNovaCarta

```
void insereNovaCarta(Gato novaCarta)
              throws SQLException
```

Assinatura do método que faz a inserção de uma nova carta do tipo Gato.

Parameters:

`novaCarta` - (Gato) nova carta a ser inserida no baralho.

Throws:

`SQLException` - Se der problema ao inicializar o banco de dados.

insereNovaCarta

```
void insereNovaCarta(LinguagensProgramacao novaCarta)
              throws SQLException
```

Assinatura do método que faz a inserção de uma nova carta do tipo LinguagensProgramacao.

Parameters:

`novaCarta` - (LinguagensProgramacao) nova carta a ser inserida no baralho.

Throws:

`SQLException` - Se der problema ao inicializar o banco de dados.

buscaTodasCartas

```
ArrayList <Carta> buscaTodasCartas()
                          throws SQLException
```

Assinatura do método que pega todas as cartas do jogo. Devolve as listas de todos os baralhos.

Returns:

('ArrayList') Lista de cartas que existem no jogo de Supertrunfo (sem ser de um tipo específico).

Throws:

`SQLException` - Se der problema ao inicializar o banco de dados.

Module com.example.supertrunfo

Package poo.trabalhofinal.supertrunfo.classes.interfaces

Class CartasRepositoryImpl<T>

java.lang.Object
poo.trabalhofinal.supertrunfo.classes.interfaces.CartasRepositoryImpl<T>

Type Parameters:

T - tipo genérico que representa o tipo de cartas que serão utilizadas.

All Implemented Interfaces:

CartasRepository<T>

```
public class CartasRepositoryImpl<T>
extends Object
implements CartasRepository<T>
```

Classe CartasRepositoryImpl

Classe que faz a conexão com o banco de dados de cartas para buscar cartas, ver cartas e inserir cartas ou modificar o banco de dados.

Implementa a interface CartasRepository, colocando o corpo dos métodos do contrato estabelecido com a interface.

Field Summary

Fields

Modifier and Type	Field	Description
private final ResourceBundle	resources	Acessa a lista recursos em "resource.properties"

Constructor Summary

Constructors

Constructor	Description
CartasRepositoryImpl()	

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description

final ArrayList <T>	buscaCartas(String jogo)	Busca o baralho de cartas de um tipo ('String') jogo no banco de dados, para as cartas poderem ser utilizadas no jogo.
ArrayList <Carta>	buscaTodasCartas()	Método que acessa o banco de dados para ver todas as cartas do jogo (todos os baralhos).
final void	insereNovaCarta (Gato novaCarta)	Método sobrecarregado para inserir uma nova carta no baralho de Gato (inserir no banco de dados).
final void	insereNovaCarta (LinguagensProgramacao novaCart	Método sobrecarregado para inserir uma nova carta no baralho de LinguagensProgramacao (inserir no banco de dados).
final void	insereNovaCarta (Personagem novaCarta)	Método para inserir uma nova carta no baralho de Personagem (inserir no banco de dados).

Methods inherited from class java.lang.Object

clone , equals , finalize , getClass , hashCode , notify , notifyAll , toString , wait , wait , wait

Field Details

resources

```
private final ResourceBundle  resources
```

Acessa a lista recursos em "resource.properties"

Constructor Details

CartasRepositoryImpl

```
public CartasRepositoryImpl()
```

Method Details

buscaCartas

```
public final ArrayList <T> buscaCartas(String  jogo)
                              throws SQLException ,
                                     JogoException
```

Busca o baralho de cartas de um tipo ('String') jogo no banco de dados, para as cartas poderem ser utilizadas no jogo.

Specified by:

`buscaCartas` in interface `CartasRepository<T>`

Parameters:

`jogo` - ('String') que representa o tipo de baralho que está sendo jogado.

Returns:

(ArrayList) lista que representa o baralho de cartas de um tipo.

Throws:

`SQLException` - Se houver falha na conexão com o banco de dados.

`JogoException` - Se a 'String' jogo fornecida for inválida (não achar um jogo/baralho daquele tipo).

insereNovaCarta

```
public final void insereNovaCarta(Personagem novaCarta)
                           throws SQLException
```

Método para inserir uma nova carta no baralho de Personagem (inserir no banco de dados).

Specified by:

`insereNovaCarta` in interface `CartasRepository<T>`

Parameters:

`novaCarta` - (Personagem) nova carta a ser inserida no baralho.

Throws:

`SQLException` - Se houver erro ao tentar conectar com o banco de dados.

insereNovaCarta

```
public final void insereNovaCarta(Gato novaCarta)
                           throws SQLException
```

Método sobrecarregado para inserir uma nova carta no baralho de Gato (inserir no banco de dados).

Specified by:

`insereNovaCarta` in interface `CartasRepository<T>`

Parameters:

`novaCarta` - (Gato) nova carta a ser inserida no baralho.

Throws:

`SQLException` - Se houver erro ao tentar conectar com o banco de dados.

insereNovaCarta

```
public final void insereNovaCarta(LinguagensProgramacao novaCarta)
                           throws SQLException
```

Método sobrecarregado para inserir uma nova carta no baralho de LinguagensProgramacao (inserir no banco de dados).

Specified by:

`insereNovaCarta` in interface `CartasRepository<T>`

Parameters:

`novaCarta` - (LinguagensProgramacao) nova carta a ser inserida no baralho.

Throws:

`SQLException` - Se houver erro ao tentar conectar com o banco de dados.

buscaTodasCartas

```
public ArrayList <Carta> buscaTodasCartas()
                                  throws SQLException
```

Método que acessa o banco de dados para ver todas as cartas do jogo (todos os baralhos). Na GUI há uma tela que permite ver todas as cartas. A ordenação das cartas é feita po *id*, contador incrementado ao inserir a carta no banco de dados.

Specified by:

`buscaTodasCartas` in interface `CartasRepository<T>`

Returns:

(ArrayList) lista de todas as cartas do jogo.

Throws:

`SQLException` - Se houver erro ao tentar conectar com o banco de dados.

Module com.example.supertrunfo
Package poo.trabalhofinal.supertrunfo.classes.interfaces

Interface JogadoresRepository<T>

Type Parameters:

`T` - tipo genérico que vai ser preenchido por um tipo de carta para o jogo.

All Known Implementing Classes:

`JogadoresRepositoryImpl`

```
public interface JogadoresRepository<T>
```

Interface JogoRepository

Interface que utiliza Generics e define o contrato (assinaturas dos métodos) que deve ser cumprido (implementação dos métodos) na classe *JogoRepositoryImpl*. Utiliza Generics porque existem 3 tipos de cartas (tipos dos baralhos) que podem ser utilizados no jogo.

Method Summary

All Methods Instance Methods Abstract Methods

Modifier and Type	Method	Description
`Jogador<T>`	`buscaJogador(String usuario, String senha)`	Assinatura do método responsável por buscar o jogador no banco de dados.
`void`	`insereNovoJogador (Jogador novoJogador)`	Assinatura do método responsável por cadastrar um novo jogador.
`void`	`updateJogadores (Jogador jogadorA, Jogador jogadorB)`	Assinatura do método responsável por atualizar os dados dos jogadores após os jogos.

Method Details

buscaJogador

```
Jogador<T> buscaJogador(String  usuario,
                        String  senha)
                 throws SQLException ,
                        UsuarioNaoEncontradoException,
                        InformacaoInvalidaException
```

Assinatura do método responsável por buscar o jogador no banco de dados.

Parameters:

`usuario` - ('String') nome do usuário.

`senha` - ('String') senha do usuário.

Returns:

(Jogador) o jogador procurado (T é o tipo de baralho para o jogo).

Throws:

`SQLException` - Se não conseguir conectar com o banco de dados.

`UsuarioNaoEncontradoException` - Se não encontrar o usuário com os dados fornecidos.

`InformacaoInvalidaException` - Se as informações fornecidas forem inválidas.

insereNovoJogador

```
void insereNovoJogador(Jogador novoJogador)
                throws SQLException
```

Assinatura do método responsável por cadastrar um novo jogador.

Parameters:

`novoJogador` - (Jogador) que será inserido (cadastrado).

Throws:

`SQLException` - Se não conseguir conectar com o banco de dados.

updateJogadores

```
void updateJogadores(Jogador jogadorA,
                     Jogador jogadorB)
              throws SQLException
```

Assinatura do método responsável por atualizar os dados dos jogadores após os jogos.

Parameters:

`jogadorA` - (Jogador) um dos jogadores para atualizar os dados após a jogada.

`jogadorB` - (Jogador) um dos jogadores para atualizar os dados após a jogada.

Throws:

`SQLException` - Se não conseguir conectar com o banco de dados.

Module com.example.supertrunfo

Package poo.trabalhofinal.supertrunfo.classes.interfaces

Class JogadoresRepositoryImpl<T>

java.lang.Object
poo.trabalhofinal.supertrunfo.classes.interfaces.JogadoresRepositoryImpl<T>

Type Parameters:

T - tipo genérico que representa o tipo de carta que foi escolhido para ser jogado pelo jogador.

All Implemented Interfaces:

JogadoresRepository<T>

```
public class JogadoresRepositoryImpl<T>
extends Object
implements JogadoresRepository<T>
```

Classe JogadoresRepositoryImpl

Classe que faz a conexão com o banco de dados de jogadores para cadastrar, logar e poder atualizar seus dados.

Implementa a interface JogadoresRepository, colocando o corpo dos métodos do contrato estabelecido com a interface.

Field Summary

Fields

Modifier and Type	Field	Description
private final ResourceBundle	resources	Acessa a lista recursos em "resource.properties"

Constructor Summary

Constructors

Constructor	Description
JogadoresRepositoryImpl()	

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description

final Jogador<T>	buscaJogador(String usuario, String senha)	Método responsável por logar o jogador.
final void	insereNovoJogador (Jogador novoJogador)	Método para cadastrar um novo jogador e inserí-lo no banco de dados.
final void	updateJogadores (Jogador jogadorA, Jogador jogadorB)	Método responsável por atualizar a pontuação de cada jogador após o jogo.

Methods inherited from class java.lang.Object

clone , equals , finalize , getClass , hashCode , notify , notifyAll , toString , wait , wait , wait

Field Details

resources

```
private final ResourceBundle  resources
```

Acessa a lista recursos em "resource.properties"

Constructor Details

JogadoresRepositoryImpl

```
public JogadoresRepositoryImpl()
```

Method Details

buscaJogador

```
public final Jogador<T> buscaJogador(String  usuario,
                                     String  senha)
                              throws SQLException ,
                                     UsuarioNaoEncontradoException,
                                     InformacaoInvalidaException
```

Método responsável por logar o jogador.

Specified by:

buscaJogador in interface JogadoresRepository<T>

Parameters:

`usuario` - ('String') nome do usuário.

`senha` - ('String') senha do usuário.

Returns:

(Jogador) jogador encontrado no banco de dados (caso os dados fornecidos para 'login' sejam válidos e existam no banco de dados).

Throws:

`SQLException` - Se houver erro ao tentar conectar com o banco de dados.

`UsuarioNaoEncontradoException` - Se o usuário não for encontrado no banco de dados.

`InformacaoInvalidaException` - Se as informações fornecidas pelo jogador forem inválidas.

insereNovoJogador

```
public final void insereNovoJogador(Jogador novoJogador)
                            throws SQLException
```

Método para cadastrar um novo jogador e inseri-lo no banco de dados. Verifica se o jogador já existe (só pode ter um usuário com um tipo de dados específico).

Specified by:

`insereNovoJogador` in interface `JogadoresRepository<T>`

Parameters:

`novoJogador` - (Jogador) que será inserido (cadastrado).

Throws:

`SQLException` - Se houver erro ao tentar conectar com o banco de dados.

updateJogadores

```
public final void updateJogadores(Jogador jogadorA,
                                  Jogador jogadorB)
                           throws SQLException
```

Método responsável por atualizar a pontuação de cada jogador após o jogo.

Specified by:

`updateJogadores` in interface `JogadoresRepository<T>`

Parameters:

`jogadorA` - (Jogador) um dos jogadores para atualizar os dados após a jogada.

`jogadorB` - (Jogador) um dos jogadores para atualizar os dados após a jogada.

Throws:

`SQLException` - Se houver erro ao tentar conectar com o banco de dados.

Module com.example.supertrunfo
Package poo.trabalhofinal.supertrunfo.classes.utils

Class Util

java.lang.Object
    poo.trabalhofinal.supertrunfo.classes.utils.Util

```
public class Util
extends Object
```

Classe Util

Classe com funcionalidades/métodos utilitários para diferentes contextos (terefas). Fornece métodos estáticos que podem ser acessados sem a necessidade de crinstanciar a classe.

Constructor Summary

Constructors

Constructor	Description
`Util()`	

Method Summary

All Methods Static Methods Concrete Methods

Modifier and Type	Method	Description
`static String`	`codificaSenha(String senha)`	Método responsável por codificar a senha.
`static Classificacao`	`stringToClassificacao(String str)`	Método para transformar uma 'String' que representa uma classificação em um tipo Classificacao(enum).

Methods inherited from class java.lang.Object

`clone` , `equals` , `finalize` , `getClass` , `hashCode` , `notify` , `notifyAll` , `toString` , `wait` , `wait` , `wait`

Constructor Details

Util

```
public Util()
```

Method Details

codificaSenha

```
public static String  codificaSenha(String  senha)
```

Método responsável por codificar a senha. Utiliza a biblioteca BCrypt para criptografar a senha.

Parameters:

`senha` - ('String') senha que o usuário digitou.

Returns:

senha codificada.

stringToClassificacao

```
public static Classificacao stringToClassificacao(String  str)
```

Método para transformar uma 'String' que representa uma classificação em um tipo Classificacao(enum).

Parameters:

`str` - ('String') que representa a classificação.

Returns:

(Classificação) classificação convertida no tipo enum.

Module com.example.supertrunfo

Package poo.trabalhofinal.supertrunfo.classes.exceptions

Class InformacaoInvalidaException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      poo.trabalhofinal.supertrunfo.classes.exceptions.InformacaoInvalidaException

All Implemented Interfaces:

`Serializable`

```
public class InformacaoInvalidaException
extends Exception
```

Classe de Exceção para informações inválidas

Classe que herda a superclasse Exception.

Lança uma exceção a ser tratada caso o usuário informe alguma informação incorreta ao programa. Usada para tratamento de erro.

See Also:

Serialized Form

Constructor Summary

Constructors

Constructor	Description
`InformacaoInvalidaException(String msg)`	Construtor da classe de Exceção.

Method Summary

Methods inherited from class java.lang.Throwable

`addSuppressed` , `fillInStackTrace` , `getCause` , `getLocalizedMessage` , `getMessage` , `getStackTrace` , `getSuppressed` , `initCause` , `printStackTrace` , `printStackTrace` , `printStackTrace` , `setStackTrace` , `toString`

Methods inherited from class java.lang.Object

`clone` , `equals` , `finalize` , `getClass` , `hashCode` , `notify` , `notifyAll` , `wait` , `wait` , `wait`

Constructor Details

InformacaoInvalidaException

```
public InformacaoInvalidaException(String  msg)
```

Construtor da classe de Exceção. Usa um dos construtores da superclasse (possui vários construtores com sobrecarga), passando a mensagem de erro ('String')

Parameters:

`msg` - ('String') menssagem associada ao erro ocorrido.

Module com.example.supertrunfo

Package poo.trabalhofinal.supertrunfo.classes.exceptions

Class JogoException

java.lang.Object
    java.lang.Throwable
        java.lang.Exception
            poo.trabalhofinal.supertrunfo.classes.exceptions.JogoException

All Implemented Interfaces:

`Serializable`

```
public class JogoException
extends Exception
```

Classe de Exceção para Jogo

Classe que herda a superclasse Exception.

Lança uma exceção a ser tratada caso ocorra algum erro ao iniciar o jogo. Usada para tratamento de erro.

See Also:

Serialized Form

Constructor Summary

Constructors

Constructor	Description
`JogoException(String msg)`	Construtor da classe de Exceção.

Method Summary

Methods inherited from class java.lang.Throwable

`addSuppressed` , `fillInStackTrace` , `getCause` , `getLocalizedMessage` , `getMessage` , `getStackTrace` , `getSuppressed` , `initCause` , `printStackTrace` , `printStackTrace` , `printStackTrace` , `setStackTrace` , `toString`

Methods inherited from class java.lang.Object

`clone` , `equals` , `finalize` , `getClass` , `hashCode` , `notify` , `notifyAll` , `wait` , `wait` , `wait`

Constructor Details

JogoException

```
public JogoException(String  msg)
```

Construtor da classe de Exceção. Usa um dos construtores da superclasse (possui vários construtores com sobrecarga), passando a mensagem de erro ('String')

Parameters:

`msg` - ('String') menssagem associada ao erro ocorrido.

Module com.example.supertrunfo

Package poo.trabalhofinal.supertrunfo.classes.exceptions

Class UsuarioNaoEncontradoException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      poo.trabalhofinal.supertrunfo.classes.exceptions.UsuarioNaoEncontradoException

All Implemented Interfaces:

`Serializable`

```
public class UsuarioNaoEncontradoException
extends Exception
```

Classe de Exceção para Usuário não encontrado

Classe que herda a superclasse Exception.

Lança uma exceção a ser tratada caso o usuário informe dados incorretos na hora de fazer login ou de cadastrar. Usada para tratamento de erro.

See Also:

Serialized Form

Constructor Summary

Constructors

Constructor	Description
`UsuarioNaoEncontradoException(String msg)`	Construtor da classe de Exceção.

Method Summary

Methods inherited from class java.lang.Throwable

`addSuppressed` , `fillInStackTrace` , `getCause` , `getLocalizedMessage` , `getMessage` , `getStackTrace` , `getSuppressed` , `initCause` , `printStackTrace` , `printStackTrace` , `printStackTrace` , `setStackTrace` , `toString`

Methods inherited from class java.lang.Object

`clone` , `equals` , `finalize` , `getClass` , `hashCode` , `notify` , `notifyAll` , `wait` , `wait` , `wait`

Constructor Details

UsuarioNaoEncontradoException

```
public UsuarioNaoEncontradoException(String  msg)
```

Construtor da classe de Exceção. Usa um dos construtores da superclasse (possui vários construtores com sobrecarga), passando a mensagem de erro ('String')

Parameters:

`msg` - ('String') menssagem associada ao erro ocorrido.

Module com.example.supertrunfo
Package poo.trabalhofinal.supertrunfo.gui

Class DBUtils

java.lang.Object
    poo.trabalhofinal.supertrunfo.gui.DBUtils

```
public class DBUtils
extends Object
```

Classe DBUtils

Classe responsável por mudar as telas da aplicação GUI.

Field Summary

Fields

Modifier and Type	Field	Description
`private static boolean`	`empatou`	(boolean) que indica se houve algum empate.
`private static final javafx.scene.image.Image`	`icon`	(Image) ícone presente nas telas.
`private static Jogo`	`jogo`	(Jogo) atributo do tipo Jogo que é passado de parâmetro em algumas mudannças de cena.
`private static Jogador`	`perdedor`	(Jogador) representa o jogador perdedor e é usado como parâmetro em algumas mudanças de cena.
`private static String`	`tipoJogo`	(‘String’) representa o tipo do jogo, passado como parâmetro em algumas mudanças de cena.
`private static Jogador`	`vencedor`	(Jogador) representa o jogador vencedor e é usado como parâmetro em algumas mudanças de cena.

Constructor Summary

Constructors

Constructor	Description
`DBUtils()`	

Method Summary

All Methods | Static Methods | Concrete Methods

Modifier and Type	Method	Description
static void	changeScene (javafx.event.ActionEvent event String fxmlFile, String title)	Método de mudança de cena (mais simples).
static void	changeScene (javafx.event.ActionEvent event String fxmlFile, String title, boolean cadastro)	Método de mudança de cena.
static void	changeScene (javafx.event.ActionEvent event String fxmlFile, String title, String tipo, boolean cadastro)	Método de mudança de cena.
static void	changeScene (javafx.scene.input.KeyEvent ev String fxmlFile, String title)	Método de mudança de cena.
static void	changeScene (javafx.scene.input.MouseEvent String fxmlFile, String title)	Método de mudança de cena.
private static void	createStage(String title, javafx.scene.Parent root, javafx.event.ActionEvent event)	Método que cria uma nova janela com base nos parâmetros fornecidos.
private static void	createStage(String title, javafx.scene.Parent root, javafx.scene.input.KeyEvent eve	Método que cria uma nova janela com base nos parâmetros fornecidos.
private static void	createStage(String title, javafx.scene.Parent root, javafx.scene.input.MouseEvent e	Método que cria uma nova janela com base nos parâmetros fornecidos.
static boolean	getEmpate()	Método para acessar o atributo de empate.
static Jogo	getJogo()	Método para acessar o atributo de jogo.

static Jogador	getPerdedor()	Método para acessar o valor do atributo jogador vencedor.
static String	getTipoJogo()	Método para acessar o atributo de tipo de jogo.
static Jogador	getVencedor()	Método para acessar o valor do atributo jogador vencedor.
static void	iniciaJogo(Jogador<? > jogadorA, Jogador<? > jogadorB, String tipo)	Métrodo para inicializar o jogo.
static void	setFimPartida(Jogador a, Jogador b, boolean empate)	Método para definir o fim da partida e definir o estado atual do jogo.

Methods inherited from class java.lang.Object

clone , equals , finalize , getClass , hashCode , notify , notifyAll , toString , wait , wait , wait

Field Details

icon

```
private static final javafx.scene.image.Image icon
```

(Image) ícone presente nas telas.

jogo

```
private static Jogo jogo
```

(Jogo) atributo do tipo Jogo que é passado de parâmetro em algumas mudannças de cena.

tipoJogo

```
private static String  tipoJogo
```

('String') representa o tipo do jogo, passado como parâmetro em algumas mudanças de cena.

vencedor

```
private static Jogador vencedor
```

(Jogador) representa o jogador vencedor e é usado como parâmetro em algumas mudanças de cena.

perdedor

```
private static Jogador perdedor
```

(Jogador) representa o jogador perdedor e é usado como parâmetro em algumas mudanças de cena.

empatou

```
private static boolean empatou
```

(boolean) que indica se houve algum empate.

Constructor Details

DBUtils

```
public DBUtils()
```

Method Details

setFimPartida

```
public static void setFimPartida(Jogador a,
                                 Jogador b,
                                 boolean empate)
```

Método para definir o fim da partida e definir o estado atual do jogo.

Parameters:

`a` - (Jogador) vencedor.

`b` - (Jogador) perdedor.

`empate` - (boolean) indica empate.

getVencedor

```
public static Jogador getVencedor()
```

Método para acessar o valor do atributo jogador vencedor.

Returns:

(Jogador) retorna o vencedor.

getPerdedor

```
public static Jogador getPerdedor()
```

Método para acessar o valor do atributo jogador vencedor.

Returns:

(Jogador) retorna o perdedor.

getEmpate

```
public static boolean getEmpate()
```

Método para acessar o atributo de empate.

Returns:

(boolean) true se houve empate ou false se não.

iniciaJogo

```
public static void iniciaJogo(Jogador<?> jogadorA,
                              Jogador<?> jogadorB,
                              String  tipo)
                       throws SQLException ,
                              JogoException
```

Métrodo para inicializar o jogo.

Parameters:

`jogadorA` - (Jogador) primeiro jogador a jogar.

`jogadorB` - (Jogador) segundo jogador a jogar.

`tipo` - (String) tipo do jogo.

Throws:

`SQLException` - Se houver problema ao conectar com banco de dados.

`JogoException` - Se houver problema ao inicializar o jogo.

getJogo

```
public static Jogo getJogo()
```

Método para acessar o atributo de jogo.

Returns:

(Jogo) true se houve empate ou false se não.

getTipoJogo

```
public static String  getTipoJogo()
```

Método para acessar o atributo de tipo de jogo.

Returns:

(String) tipo do jogo.

changeScene

```
public static void changeScene(javafx.event.ActionEvent event,
                               String  fxmlFile,
                               String  title)
```

Método de mudança de cena (mais simples).

É declarada uma variável root (Parent) e inicializada como null. Essa variável será usada para armazenar o conteúdo raiz da nova cena.

O bloco try-catch captura qualquer exceção que pode acontecer ao abrir o FXML.

Parameters:

`event` - (ActionEvent) Botão pressionado.

`fxmlFile` - (‘String’) Nome do arquivo fxml.

`title` - ('String') Titulo da cena.

changeScene

```
public static void changeScene(javafx.scene.input.KeyEvent event,
                               String  fxmlFile,
                               String  title)
```

Método de mudança de cena.

É declarada uma variável root (Parent) e inicializada como null. Essa variável será usada para armazenar o conteúdo raiz da nova cena.

O bloco try-catch captura qualquer exceção que pode acontecer ao abrir o FXML.

A diferença do outro método, é que trata de um evento de apertar uma tecla do teclado.

Parameters:

`event` - (KeyEvent) tecla pressionada.

`fxmlFile` - (‘String’) Nome do arquivo fxml.

`title` - ('String') Titulo da cena.

changeScene

```
public static void changeScene(javafx.event.ActionEvent event,
                               String  fxmlFile,
                               String  title,
                               String  tipo,
                               boolean cadastro)
```

Método de mudança de cena.

É declarada uma variável root (Parent) e inicializada como null. Essa variável será usada para armazenar o conteúdo raiz da nova cena.

O bloco try-catch captura qualquer exceção que pode acontecer ao abrir o FXML.

Recebe mais uma String que representa o tipo do jogo escolhido na tela anterior e um boolean que indica se a tela que será aberta agora será a de cadastro de cartas ou a tela de login para jogar.

Parameters:

`event` - (ActionEvent) Botão pressionado.

`fxmlFile` - (‘String’) Nome do arquivo fxml.

`title` - ('String') Titulo da cena.

`tipo` - ('String') Tipo do jogo.

`cadastro` - (boolean) true se for para cadastrar cartas e false se for para login.

changeScene

```
public static void changeScene(javafx.event.ActionEvent event,
                               String  fxmlFile,
                               String  title,
                               boolean cadastro)
```

Método de mudança de cena.

É declarada uma variável root (Parent) e inicializada como null. Essa variável será usada para armazenar o conteúdo raiz da nova cena.

O bloco try-catch captura qualquer exceção que pode acontecer ao abrir o FXML.

Recebe ainda um parâmetro do tipo cadastro, indicando se vai ou não para tela de cadastro.

Parameters:

`event` - (ActionEvent) Botão pressionado.

`fxmlFile` - (‘String’) Nome do arquivo fxml.

`title` - ('String') Titulo da cena.

`cadastro` - (boolean) true se for para cadastro e false se não for.

changeScene

```
public static void changeScene(javafx.scene.input.MouseEvent event,
                               String  fxmlFile,
                               String  title)
```

Método de mudança de cena.

É declarada uma variável root (Parent) e inicializada como null. Essa variável será usada para armazenar o conteúdo raiz da nova cena.

O bloco try-catch captura qualquer exceção que pode acontecer ao abrir o FXML.

Trata um evento de mouse (caso da tela inicial).

Parameters:

`event` - (MouseEvent) click do mouse.

`fxmlFile` - ('String') Nome do arquivo fxml.

`title` - ('String') Titulo da cena.

createStage

```
private static void createStage(String  title,
                                javafx.scene.Parent root,
                                javafx.event.ActionEvent event)
```

Método que cria uma nova janela com base nos parâmetros fornecidos.

Coloca um título na janela, não deixa ser redimensionável e define o tamanho da tela.

Parameters:

`title` - (String) Título da tela.

`root` - (Parent) armazena o conteúdo da raiz da nova cena.

`event` - (ActionEvent) evento para mudança de tela.

createStage

```
private static void createStage(String  title,
                                javafx.scene.Parent root,
                                javafx.scene.input.KeyEvent event)
```

Método que cria uma nova janela com base nos parâmetros fornecidos.

Coloca um título na janela, não deixa ser redimensionável e define o tamanho da tela.

A diferença do outro método é que esse trata de um KeyEvent (evento do teclado).

Parameters:

`title` - (String) Título da tela.

`root` - (Parent) armazena o conteúdo da raiz da nova cena.

`event` - (KeyEvent) evento para mudança de tela.

createStage

```
private static void createStage(String  title,
                                javafx.scene.Parent root,
                                javafx.scene.input.MouseEvent event)
```

Método que cria uma nova janela com base nos parâmetros fornecidos.

Coloca um título na janela, não deixa ser redimensionável e define o tamanho da tela.

A diferença do outro método é que esse trata de um MouseEvent (evento do mouse).

Parameters:

`title` - (String) Título da tela.

`root` - (Parent) armazena o conteúdo da raiz da nova cena.

`event` - (MouseEvent) evento para mudança de tela.

Module com.example.supertrunfo
Package poo.trabalhofinal.supertrunfo.gui.controllers

Class CadastroCartaController

java.lang.Object
poo.trabalhofinal.supertrunfo.gui.controllers.CadastroCartaController

All Implemented Interfaces:

`javafx.fxml.Initializable`

```
public class CadastroCartaController
extends Object
implements javafx.fxml.Initializable
```

Classe CadastroCartaController

Classe responsável por intermediar a relação entre a interface (GUI) da *Tela de cadastro de carta* e o programa.

Recebe as solicitações da interface e trata os eventos de acordo com o esperado no programa. Permite que o usuário faça cadastro de uma nova carta para poder jogar com ela depois, armazenando seus dados no banco de dados de cartas.

Implementa a interface *Initializable* do JavaFX, que define a assinatura do método de inicialização de um controller da tela.

Field Summary

Fields

Modifier and Type	Field	Description
`javafx.scene.control.Button`	`adicionar`	Elemento FXML (botão) que ao ser clicado permite que a carta seja cadastrada no banco de dados.
`javafx.scene.control.Label`	`alerta`	Elemento FXML que coloca uma mensagem de erro na tela caso haja algum problema na hora de cadastrar a carta.
`javafx.scene.control.TextField`	`caracteristica1`	Elemento FXML que pega a primeira característica da carta por meio da GUI.
`javafx.scene.control.TextField`	`caracteristica2`	Elemento FXML que pega a segunda característica da carta por meio da GUI.
`javafx.scene.control.TextField`	`caracteristica3`	Elemento FXML que pega a terceira característica da carta por meio da GUI.

javafx.scene.control.TextField	**caracteristica4**	Elemento FXML que pega a quarta característica da carta por meio da GUI.
javafx.scene.control.TextField	**caracteristica5**	Elemento FXML que pega a quinta característica da carta por meio da GUI.
javafx.scene.control.TextField	**classificacao**	Elemento FXML que pega a classificação da carta por meio da GUI.
javafx.scene.control.TextField	**imagem**	Elemento FXML que pega a url da imagem da carta por meio da GUI.
javafx.scene.control.TextField	**nome**	Elemento FXML que pega o nome da carta por meio da GUI.
(package private) String	**tipo**	('String') atributo privado que indica o tipo de jogo (baralho).
javafx.scene.control.Button	**voltar**	Elemento FXML (botão) que ao ser clicado permite que o usuário retorne ao menu.

Constructor Summary

Constructors

Constructor	Description
CadastroCartaController()	

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
void	**adicionarCarta**(javafx.event.ActionEvent event, javafx.scene.input.KeyEvent keyEv	Método para cadastrar as cartas, verificando o tipo de carta escolhido e adicionando a nova carta no banco de dados.
void	**initialize**(URL url, ResourceBundle resourceBundle)	Método sobrescrito oriundo da interface *Initializable*.
void	**setDados**(String tipo)	Método que modifca o texto presente no prompt do Text Field de acordo com o tipo da carta, colcoando cada característica do respectivo tipo de carta.
private void	**validaPreenchidos**()	Método que valida se todos os campos foram preenchidos.

`private void validaValores()`	Método que valida se os dados inseridos nos campos estão válidos.

Methods inherited from class java.lang.Object

`clone` , `equals` , `finalize` , `getClass` , `hashCode` , `notify` , `notifyAll` , `toString` , `wait` , `wait` , `wait`

Field Details

nome

`public javafx.scene.control.TextField nome`

Elemento FXML que pega o nome da carta por meio da GUI.

classificacao

`public javafx.scene.control.TextField classificacao`

Elemento FXML que pega a classificação da carta por meio da GUI.

caracteristica1

`public javafx.scene.control.TextField caracteristica1`

Elemento FXML que pega a primeira característica da carta por meio da GUI.

caracteristica2

`public javafx.scene.control.TextField caracteristica2`

Elemento FXML que pega a segunda característica da carta por meio da GUI.

caracteristica3

`public javafx.scene.control.TextField caracteristica3`

Elemento FXML que pega a terceira característica da carta por meio da GUI.

caracteristica4

`public javafx.scene.control.TextField caracteristica4`

Elemento FXML que pega a quarta característica da carta por meio da GUI.

caracteristica5

```
public javafx.scene.control.TextField caracteristica5
```

Elemento FXML que pega a quinta característica da carta por meio da GUI.

imagem

```
public javafx.scene.control.TextField imagem
```

Elemento FXML que pega a url da imagem da carta por meio da GUI.

alerta

```
public javafx.scene.control.Label alerta
```

Elemento FXML que coloca uma mensagem de erro na tela caso haja algum problema na hora de cadastrar a carta. Mostra que o usuário conseguiu cadastrar também.

adicionar

```
public javafx.scene.control.Button adicionar
```

Elemento FXML (botão) que ao ser clicado permite que a carta seja cadastrada no banco de dados.

voltar

```
public javafx.scene.control.Button voltar
```

Elemento FXML (botão) que ao ser clicado permite que o usuário retorne ao menu.

tipo

```
String  tipo
```

('String') atributo privado que indica o tipo de jogo (baralho).

Constructor Details

CadastroCartaController

```
public CadastroCartaController()
```

Method Details

setDados

```
public void setDados(String  tipo)
```

Método que modifca o texto presente no prompt do Text Field de acordo com o tipo da carta, colcoando cada característica do respectivo tipo de carta.

Parameters:

`tipo` - ('String') que representa o tipo de jogo (baralho).

initialize

```
public void initialize(URL  url,
                       ResourceBundle  resourceBundle)
```

Método sobrescrito oriundo da interface *Initializable*.

Método que verifica o tipo e chama o método de conexão com banco da dados de carta, presente na classe CartasRepositoryImpl. Coloca um alerta na tela caso tenha havido algum erro ou mensagem de cadastrado com sucesso.

Chama um método que verifica se todos os campos estão preenchidos, caso não estejam e haja erro, limpam o que já foi preenchido para usuário preencher novamente.

Specified by:

`initialize` in interface `javafx.fxml.Initializable`

Parameters:

`url` - (URL) do elemento fxml que está sendo carregado.

`resourceBundle` - (ResourceBundle) é fornecido como convenção para permitir o acesso a recursos adicionais.

adicionarCarta

```
public void adicionarCarta(javafx.event.ActionEvent event,
                           javafx.scene.input.KeyEvent keyEvent)
```

Método para cadastrar as cartas, verificando o tipo de carta escolhido e adicionando a nova carta no banco de dados.

Parameters:

`event` - (ActionEvent) botão clicado (cadastrar).

`keyEvent` - (KeyEvent) quando aperta ENTER também faz o cadastro.

validaPreenchidos

```
private void validaPreenchidos()
                       throws InformacaoInvalidaException
```

Método que valida se todos os campos foram preenchidos.

Throws:

`InformacaoInvalidaException` - Caso um dos campos não tenham sido preenchidos.

validaValores

```
private void validaValores()
                    throws InformacaoInvalidaException
```

Método que valida se os dados inseridos nos campos estão válidos.

Throws:

`InformacaoInvalidaException` - Caso um dos dados sejam inválidos.

Module com.example.supertrunfo
Package poo.trabalhofinal.supertrunfo.gui.controllers

Class CadastroUsuarioController

java.lang.Object
    poo.trabalhofinal.supertrunfo.gui.controllers.CadastroUsuarioController

All Implemented Interfaces:

`javafx.fxml.Initializable`

```
public class CadastroUsuarioController
extends Object
implements javafx.fxml.Initializable
```

Classe CadastroUsuarioController

Classe responsável por intermediar a relação entre a interface (GUI) da *Tela de cadastro de usuário* e o programa.

Recebe as solicitações da interface e trata os eventos de acordo com o esperado no programa. Permite que o usuário faça cadastro para poder jogar depois, armazenando seus dados no banco de dados de jogadores.

Implementa a interface *Initializable* do JavaFX, que define a assinatura do método de inicialização de um controller da tela.

Field Summary

Fields

Modifier and Type	Field	Description
`javafx.scene.control.Label`	`alerta`	Elemento FXML que coloca uma mensagem de erro na tela caso haja algum problema na hora de cadastrar.
`javafx.scene.control.Button`	`cadastrar`	Elemento FXML (botão) que, ao ser clicado, permite que o usuário faça cadastro.
`javafx.scene.control.TextField`	`nome`	Elemento FXML que pega o nome do usuário por meio da GUI.
`javafx.scene.control.PasswordFi`	`senha`	Elemento FXML que pega a senha do usuário por meio da GUI.
`javafx.scene.control.Button`	`voltar`	Elemento FXML (botão) que, ao ser clicado, permite que o usuário volte para o menu.

Constructor Summary

Constructors

Constructor	Description
`CadastroUsuarioController()`	

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
private void	`cadastrarUsuario (javafx.event.ActionEvent event, javafx.scene.input.KeyEvent keyEv`	Método para cadastrar o usuário, adicionando-no no banco de dados.
void	`initialize(URL url, ResourceBundle resourceBundle)`	Método sobrescrito oriundo da interface *Initializable*.
private void	`validaPreenchidos()`	Método que valida se todos os campos foram preenchidos.

Methods inherited from class java.lang.Object

`clone` , `equals` , `finalize` , `getClass` , `hashCode` , `notify` , `notifyAll` , `toString` , `wait` , `wait` , `wait`

Field Details

nome

```
public javafx.scene.control.TextField nome
```

Elemento FXML que pega o nome do usuário por meio da GUI.

senha

```
public javafx.scene.control.PasswordField senha
```

Elemento FXML que pega a senha do usuário por meio da GUI.

cadastrar

```
public javafx.scene.control.Button cadastrar
```

Elemento FXML (botão) que, ao ser clicado, permite que o usuário faça cadastro.

voltar

```
public javafx.scene.control.Button voltar
```

Elemento FXML (botão) que, ao ser clicado, permite que o usuário volte para o menu.

alerta

```
public javafx.scene.control.Label alerta
```

Elemento FXML que coloca uma mensagem de erro na tela caso haja algum problema na hora de cadastrar. Mostra que o usuário conseguiu cadastrar também.

Constructor Details

CadastroUsuarioController

```
public CadastroUsuarioController()
```

Method Details

initialize

```
public void initialize(URL  url,
                       ResourceBundle  resourceBundle)
```

Método sobrescrito oriundo da interface *Initializable*.

Recebe as solicitações da interface e trata os eventos de acordo com o esperado no programa. Conecta por meio da classe de JogadoresRepositoryImpl ao banco de dados de jogadores, para inserir novo jogador. Coloca uma mensagem de alerta na tela caso haja algum erro no processo. Alerta também coloca uma mensagem indicando que o jogador foi cadastrado com sucesso. Muda para a tela de menu após o cadastro.

Specified by:

`initialize` in interface `javafx.fxml.Initializable`

Parameters:

`url` - (URL) do elemento fxml que está sendo carregado.

`resourceBundle` - (ResourceBundle) é fornecido como convenção para permitir o acesso a recursos adicionais.

cadastrarUsuario

```
private void cadastrarUsuario(javafx.event.ActionEvent event,
                              javafx.scene.input.KeyEvent keyEvent)
```

Método para cadastrar o usuário, adicionando-no no banco de dados.

Parameters:

`event` - (ActionEvent) botão clicado (cadastrar).

`keyEvent` - (KeyEvent) quando aperta ENTER também faz o cadastro.

validaPreenchidos

```
private void validaPreenchidos()
                       throws InformacaoInvalidaException
```

Método que valida se todos os campos foram preenchidos.

Throws:

`InformacaoInvalidaException` - Caso um dos campos não tenham sido preenchidos.

Module com.example.supertrunfo
Package poo.trabalhofinal.supertrunfo.gui.controllers

Class JogoController

java.lang.Object
    poo.trabalhofinal.supertrunfo.gui.controllers.JogoController

All Implemented Interfaces:

`javafx.fxml.Initializable`

```
public class JogoController
extends Object
implements javafx.fxml.Initializable
```

Classe JogoController

Classe responsável por intermediar a relação entre a interface (GUI) da *Tela de jogo* e o programa.

Recebe as solicitações da interface e trata os eventos de acordo com o esperado no programa. Permite que os jogadores reecebam suas cartas e possam jogar, modificando seus dados para posteriormente eles possam ser atualizados no banco de dados (na tela de vencedor).

Implementa a interface *Initializable* do JavaFX, que define a assinatura do método de inicialização de um controller da tela.

Field Summary

Fields

Modifier and Type	Field	Description
`javafx.scene.control.Label`	`alertaA`	Elemento FXML que representa o alerta para o jogador A
`javafx.scene.control.Label`	`alertaB`	Elemento FXML que representa o alerta para o jogador A
`javafx.scene.control.Button`	`b_caracteristicaA1`	Elemento FXML que representa o botão associado a essa característica (primeira) da carta.
`javafx.scene.control.Button`	`b_caracteristicaA2`	Elemento FXML que representa o botão associado a essa característica (segunda) da carta.
`javafx.scene.control.Button`	`b_caracteristicaA3`	Elemento FXML que representa o botão associado a essa característica (terceira) da carta.
`javafx.scene.control.Button`	`b_caracteristicaA4`	Elemento FXML que representa o botão associado a essa característica

		(quarta) da carta.
javafx.scene.control.Button	**b_caracteristicaA5**	Elemento FXML que representa o botão associado a essa característica (quinta) da carta.
javafx.scene.control.Button	**b_caracteristicaB1**	Elemento FXML que representa o botão associado a essa característica (primeira) da carta.
javafx.scene.control.Button	**b_caracteristicaB2**	Elemento FXML que representa o botão associado a essa característica (segunda) da carta.
javafx.scene.control.Button	**b_caracteristicaB3**	Elemento FXML que representa o botão associado a essa característica (terceira) da carta.
javafx.scene.control.Button	**b_caracteristicaB4**	Elemento FXML que representa o botão associado a essa característica (quarta) da carta.
javafx.scene.control.Button	**b_caracteristicaB5**	Elemento FXML que representa o botão associado a essa característica (quinta) da carta.
javafx.scene.control.Button	**b_superA**	Elemento FXML que representa o botão de supertrunfo presentes em cartas supertrunfo que o jogador A pegar.
javafx.scene.control.Button	**b_superB**	Elemento FXML que representa o botão de supertrunfo presentes em cartas supertrunfo que o jogador B pegar.
javafx.scene.control.Label	**caracteristicaA1**	Elemento FXML que representa o label da primeira característica na carta do jogador A (escreve qual é a característica).
javafx.scene.control.Label	**caracteristicaA2**	Elemento FXML que representa o label da segunda característica na carta do jogador A (escreve qual é a característica).
javafx.scene.control.Label	**caracteristicaA3**	Elemento FXML que representa o label da terceira característica na carta do jogador A (escreve qual é a característica).
javafx.scene.control.Label	**caracteristicaA4**	Elemento FXML que representa o label da quarta característica na carta do jogador A (escreve qual é a característica).

javafx.scene.control.Label	**caracteristicaA5**	Elemento FXML que representa o label da quinta característica na carta do jogador A (escreve qual é a característica).
javafx.scene.control.Label	**caracteristicaB1**	Elemento FXML que representa o label da primeira característica na carta do jogador B (escreve qual é a característica).
javafx.scene.control.Label	**caracteristicaB2**	Elemento FXML que representa o label da segunda característica na carta do jogador B (escreve qual é a característica).
javafx.scene.control.Label	**caracteristicaB3**	Elemento FXML que representa o label da terceira característica na carta do jogador B (escreve qual é a característica).
javafx.scene.control.Label	**caracteristicaB4**	Elemento FXML que representa o label da quarta característica na carta do jogador B (escreve qual é a característica).
javafx.scene.control.Label	**caracteristicaB5**	Elemento FXML que representa o label da quinta característica na carta do jogador B (escreve qual é a característica).
javafx.scene.control.Label	**classificacaoA**	Elemento FXML que representa a classificação na carta do jogador A.
javafx.scene.control.Label	**classificacaoB**	Elemento FXML que representa a classificação na carta do jogador B.
javafx.scene.control.Button	**empatar**	Elemento FXML do botão de empatar a partida.
javafx.scene.control.Button	**encerrar**	Elemento FXML do botão de encerrar a partida como está.
javafx.scene.image.ImageView	**fundo**	Elemento FXML que coloca a imagem da tela do jogo.
(package private) final javafx.scene.image.Image	**fundoA**	Imagem para colocar na tela do jogo quando o jogador A joga.
(package private) final javafx.scene.image.Image	**fundoB**	Imagem para colocar na tela do jogo quando o jogador B joga.
javafx.scene.image.ImageView	**imagemA**	Elemento FXML que representa a imagem na carta do jogador A.

javafx.scene.image.ImageView	**imagemB**	Elemento FXML que representa a imagem na carta do jogador B.
private static final **Jogo**	**jogo**	Constante jogo, que representa o jogo que está sendo jogado (a partida inteira).
javafx.scene.control.Label	**nomeA**	Elemento FXML que representa o nome na carta do jogador A (nome do personagem, gato ou linguagem na carta).
javafx.scene.control.Label	**nomeB**	Elemento FXML que representa o nome na carta do jogador B (nome do personagem, gato ou linguagem na carta).
javafx.scene.control.Label	**qtdCartasA**	Elemento FXML que representa a quantidade de cartas do jogador A.
javafx.scene.control.Label	**qtdCartasB**	Elemento FXML que representa a quantidade de cartas do jogador B.
javafx.scene.control.Button	**render**	Elemento FXML do botão de se render.
private int	**rodada**	Contador da rodada que está → ver de quem é a vez de jogar.
javafx.scene.image.ImageView	**superA**	Elemento FXML que representa a imagem de uma carta supertrunfo.
javafx.scene.image.ImageView	**superB**	Elemento FXML que representa a imagem de uma carta supertrunfo.
private static final **String**	**tipo**	Constante que representa o tipo do jogo.
private **Carta**	**topoA**	Carta do topo do jogador A (carta apresentada na tela).
private **Carta**	**topoB**	Carta do topo do jogador B (carta apresentada na tela).
javafx.scene.control.Label	**turno**	Elemento FXML que coloca o turno na tela.
private int	**ultimoA**	Guarda a última característica jogada por A
private int	**ultimoB**	Guarda a última característica jogada por B
(package private) final **URL**	**urlFundoA**	URL do fundo quando o jogador A joga.

(package private) final URL	urlFundoB	URL do fundo quando o jogador B joga.
javafx.scene.control.Label	usuarioA	Elemento FXML que representa o nome do jogador A.
javafx.scene.control.Label	usuarioB	Elemento FXML que representa o nome do jogador B.
javafx.scene.control.Label	valorA1	Elemento FXML que representa o valor associado a primeira característica na carta do jogador A.
javafx.scene.control.Label	valorA2	Elemento FXML que representa o valor associado a segunda característica na carta do jogador A.
javafx.scene.control.Label	valorA3	Elemento FXML que representa o valor associado a terceira característica na carta do jogador A.
javafx.scene.control.Label	valorA4	Elemento FXML que representa o valor associado a quarta característica na carta do jogador A.
javafx.scene.control.Label	valorA5	Elemento FXML que representa o valor associado a quinta característica na carta do jogador A.
javafx.scene.control.Label	valorB1	Elemento FXML que representa o valor associado a primeira característica na carta do jogador B.
javafx.scene.control.Label	valorB2	Elemento FXML que representa o valor associado a segunda característica na carta do jogador B.
javafx.scene.control.Label	valorB3	Elemento FXML que representa o valor associado a terceira característica na carta do jogador B.
javafx.scene.control.Label	valorB4	Elemento FXML que representa o valor associado a quarta característica na carta do jogador B.
javafx.scene.control.Label	valorB5	Elemento FXML que representa o valor associado a quinta característica na carta do jogador B.

Constructor Summary

Constructors

Constructor	Description
`JogoController()`	

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
void	`initialize(URL location, ResourceBundle resources)`	Método sobrescrito oriundo da interface *Initializable*.
private void	`mostrarCartaA(boolean mostraB)`	Função responsável por mostrar a carta do jogador A na sua rodada.
private void	`mostrarCartaB(boolean mostraA)`	Função para mmostrar a carta de B na sua rodada, escondendo a carta do jogador A.
private void	`pegaTopo()`	Função que pega a carta do topo para ela ser mostrada na tela.
private void	`setLabel()`	Função responsável por verificar o tipo do jogo e modificar as labels conforme as características presentes nesse tipo de carta.
private void	`setTurno (javafx.event.ActionEvent event)`	Verifica se o turno é de A ou de B.
private Jogador	`vencedorRodadaC1()`	Verifica Qual o jogador vencedor, caso a característica escolhida tenha sido a 1.
private Jogador	`vencedorRodadaC2()`	Verifica Qual o jogador vencedor, caso a característica escolhida tenha sido a 2.
private Jogador	`vencedorRodadaC3()`	Verifica Qual o jogador vencedor, caso a característica escolhida tenha sido a 3.
private Jogador	`vencedorRodadaC4()`	Verifica Qual o jogador vencedor, caso a característica escolhida tenha sido a 4.
private Jogador	`vencedorRodadaC5()`	Verifica Qual o jogador vencedor, caso a característica escolhida tenha sido a 5.
private void	`verificaVencedor (javafx.event.ActionEvent event, Jogador vencedor)`	Método responsável por verificar o vencedor da rodada.

Methods inherited from class java.lang.Object

`clone` , `equals` , `finalize` , `getClass` , `hashCode` , `notify` , `notifyAll` , `toString` , `wait` , `wait` , `wait`

Field Details

imagemA

```
public javafx.scene.image.ImageView imagemA
```

Elemento FXML que representa a imagem na carta do jogador A.

classificacaoA

```
public javafx.scene.control.Label classificacaoA
```

Elemento FXML que representa a classificação na carta do jogador A.

nomeA

```
public javafx.scene.control.Label nomeA
```

Elemento FXML que representa o nome na carta do jogador A (nome do personagem, gato ou linguagem na carta).

qtdCartasA

```
public javafx.scene.control.Label qtdCartasA
```

Elemento FXML que representa a quantidade de cartas do jogador A.

usuarioA

```
public javafx.scene.control.Label usuarioA
```

Elemento FXML que representa o nome do jogador A.

caracteristicaA1

```
public javafx.scene.control.Label caracteristicaA1
```

Elemento FXML que representa o label da primeira característica na carta do jogador A (escreve qual é a característica).

valorA1

```
public javafx.scene.control.Label valorA1
```

Elemento FXML que representa o valor associado a primeira característica na carta do jogador A.

b_caracteristicaA1

```
public javafx.scene.control.Button b_caracteristicaA1
```

Elemento FXML que representa o botão associado a essa característica (primeira) da carta.

caracteristicaA2

```
public javafx.scene.control.Label caracteristicaA2
```

Elemento FXML que representa o label da segunda característica na carta do jogador A (escreve qual é a característica).

valorA2

```
public javafx.scene.control.Label valorA2
```

Elemento FXML que representa o valor associado a segunda característica na carta do jogador A.

b_caracteristicaA2

```
public javafx.scene.control.Button b_caracteristicaA2
```

Elemento FXML que representa o botão associado a essa característica (segunda) da carta.

caracteristicaA3

```
public javafx.scene.control.Label caracteristicaA3
```

Elemento FXML que representa o label da terceira característica na carta do jogador A (escreve qual é a característica).

valorA3

```
public javafx.scene.control.Label valorA3
```

Elemento FXML que representa o valor associado a terceira característica na carta do jogador A.

b_caracteristicaA3

```
public javafx.scene.control.Button b_caracteristicaA3
```

Elemento FXML que representa o botão associado a essa característica (terceira) da carta.

caracteristicaA4

```
public javafx.scene.control.Label caracteristicaA4
```

Elemento FXML que representa o label da quarta característica na carta do jogador A (escreve qual é a característica).

valorA4

```
public javafx.scene.control.Label valorA4
```

Elemento FXML que representa o valor associado a quarta característica na carta do jogador A.

b_caracteristicaA4

```
public javafx.scene.control.Button b_caracteristicaA4
```

Elemento FXML que representa o botão associado a essa característica (quarta) da carta.

caracteristicaA5

```
public javafx.scene.control.Label caracteristicaA5
```

Elemento FXML que representa o label da quinta característica na carta do jogador A (escreve qual é a característica).

valorA5

```
public javafx.scene.control.Label valorA5
```

Elemento FXML que representa o valor associado a quinta característica na carta do jogador A.

b_caracteristicaA5

```
public javafx.scene.control.Button b_caracteristicaA5
```

Elemento FXML que representa o botão associado a essa característica (quinta) da carta.

b_superA

```
public javafx.scene.control.Button b_superA
```

Elemento FXML que representa o botão de supertrunfo presentes em cartas supertrunfo que o jogador A pegar.

superA

```
public javafx.scene.image.ImageView superA
```

Elemento FXML que representa a imagem de uma carta supertrunfo.

alertaA

```
public javafx.scene.control.Label alertaA
```

Elemento FXML que representa o alerta para o jogador A

imagemB

```
public javafx.scene.image.ImageView imagemB
```

Elemento FXML que representa a imagem na carta do jogador B.

classificacaoB

```
public javafx.scene.control.Label classificacaoB
```

Elemento FXML que representa a classificação na carta do jogador B.

nomeB

```
public javafx.scene.control.Label nomeB
```

Elemento FXML que representa o nome na carta do jogador B (nome do personagem, gato ou linguagem na carta).

qtdCartasB

```
public javafx.scene.control.Label qtdCartasB
```

Elemento FXML que representa a quantidade de cartas do jogador B.

usuarioB

```
public javafx.scene.control.Label usuarioB
```

Elemento FXML que representa o nome do jogador B.

caracteristicaB1

```
public javafx.scene.control.Label caracteristicaB1
```

Elemento FXML que representa o label da primeira característica na carta do jogador B (escreve qual é a característica).

valorB1

```
public javafx.scene.control.Label valorB1
```

Elemento FXML que representa o valor associado a primeira característica na carta do jogador B.

b_caracteristicaB1

```
public javafx.scene.control.Button b_caracteristicaB1
```

Elemento FXML que representa o botão associado a essa característica (primeira) da carta.

caracteristicaB2

```
public javafx.scene.control.Label caracteristicaB2
```

Elemento FXML que representa o label da segunda característica na carta do jogador B (escreve qual é a característica).

valorB2

```
public javafx.scene.control.Label valorB2
```

Elemento FXML que representa o valor associado a segunda característica na carta do jogador B.

b_caracteristicaB2

```
public javafx.scene.control.Button b_caracteristicaB2
```

Elemento FXML que representa o botão associado a essa característica (segunda) da carta.

caracteristicaB3

```
public javafx.scene.control.Label caracteristicaB3
```

Elemento FXML que representa o label da terceira característica na carta do jogador B (escreve qual é a característica).

valorB3

```
public javafx.scene.control.Label valorB3
```

Elemento FXML que representa o valor associado a terceira característica na carta do jogador B.

b_caracteristicaB3

```
public javafx.scene.control.Button b_caracteristicaB3
```

Elemento FXML que representa o botão associado a essa característica (terceira) da carta.

caracteristicaB4

```
public javafx.scene.control.Label caracteristicaB4
```

Elemento FXML que representa o label da quarta característica na carta do jogador B (escreve qual é a característica).

valorB4

```
public javafx.scene.control.Label valorB4
```

Elemento FXML que representa o valor associado a quarta característica na carta do jogador B.

b_caracteristicaB4

```
public javafx.scene.control.Button b_caracteristicaB4
```

Elemento FXML que representa o botão associado a essa característica (quarta) da carta.

caracteristicaB5

```
public javafx.scene.control.Label caracteristicaB5
```

Elemento FXML que representa o label da quinta característica na carta do jogador B (escreve qual é a característica).

valorB5

```
public javafx.scene.control.Label valorB5
```

Elemento FXML que representa o valor associado a quinta característica na carta do jogador B.

b_caracteristicaB5

```
public javafx.scene.control.Button b_caracteristicaB5
```

Elemento FXML que representa o botão associado a essa característica (quinta) da carta.

b_superB

```
public javafx.scene.control.Button b_superB
```

Elemento FXML que representa o botão de supertrunfo presentes em cartas supertrunfo que o jogador B pegar.

superB

```
public javafx.scene.image.ImageView superB
```

Elemento FXML que representa a imagem de uma carta supertrunfo.

alertaB

```
public javafx.scene.control.Label alertaB
```

Elemento FXML que representa o alerta para o jogador A

turno

```
public javafx.scene.control.Label turno
```

Elemento FXML que coloca o turno na tela.

fundo

```
public javafx.scene.image.ImageView fundo
```

Elemento FXML que coloca a imagem da tela do jogo.

render

```
public javafx.scene.control.Button render
```

Elemento FXML do botão de se render.

encerrar

```
public javafx.scene.control.Button encerrar
```

Elemento FXML do botão de encerrar a partida como está.

empatar

```
public javafx.scene.control.Button empatar
```

Elemento FXML do botão de empatar a partida.

urlFundoA

```
final URL  urlFundoA
```

URL do fundo quando o jogador A joga.

fundoA

```
final javafx.scene.image.Image fundoA
```

Imagem para colocar na tela do jogo quando o jogador A joga.

urlFundoB

```
final URL  urlFundoB
```

URL do fundo quando o jogador B joga.

fundoB

```
final javafx.scene.image.Image fundoB
```

Imagem para colocar na tela do jogo quando o jogador B joga.

tipo

```
private static final String  tipo
```

Constante que representa o tipo do jogo.

jogo

```
private static final Jogo jogo
```

Constante jogo, que representa o jogo que está sendo jogado (a partida inteira).

topoA

```
private Carta topoA
```

Carta do topo do jogador A (carta apresentada na tela).

topoB

```
private Carta topoB
```

Carta do topo do jogador B (carta apresentada na tela).

rodada

```
private int rodada
```

Contador da rodada que está → ver de quem é a vez de jogar.

ultimoA

```
private int ultimoA
```

Guarda a última característica jogada por A

ultimoB

```
private int ultimoB
```

Guarda a última característica jogada por B

Constructor Details

JogoController

```
public JogoController()
```

Method Details

initialize

```
public void initialize(URL  location,
                       ResourceBundle  resources)
```

Método sobrescrito oriundo da interface *Initializable*.

Apresenta as características da carta na tela e mostra a carta do jogador que está jogando a partir de outros métodos externos que são chamados.

Define o que é feito ao jogador apertar em um dos botões da característica que ele quiser jogar. Verifica os valores das características das cartas do jogador A e do jogador B. Se o jogador perder na rodada dele, ele perde mais pontos que o normal.

Verifica a rodada e qual jogador pode escolher as características na rodada.

Um ponto importante é que uma mesma característica não pode ser jogada e escolhida duas vezes seguidas.

Specified by:

`initialize` in interface `javafx.fxml.Initializable`

Parameters:

`location` - (URL) do elemento fxml que está sendo carregado.

`resources` - (ResourceBundle) é fornecido como convenção para permitir o acesso a recursos adicionais.

setLabel

```
private void setLabel()
```

Função responsável por verificar o tipo do jogo e modificar as labels conforme as características presentes nesse tipo de carta.

pegaTopo

```
private void pegaTopo()
```

Função que pega a carta do topo para ela ser mostrada na tela.

mostrarCartaA

```
private void mostrarCartaA(boolean mostraB)
```

Função responsável por mostrar a carta do jogador A na sua rodada. Tampando a carta do topo de B para que o jogador A não veja a carta.

Parameters:

`mostraB` - (boolean) que indica que a carta B deve ser tampada e a carta A mostrada.

mostrarCartaB

```
private void mostrarCartaB(boolean mostraA)
```

Função para mmostrar a carta de B na sua rodada, escondendo a carta do jogador A.

Parameters:

`mostraA` - (boolean) que indica que a carta A deve ser escondida na partida do adversário.

setTurno

```
private void setTurno(javafx.event.ActionEvent event)
```

Verifica se o turno é de A ou de B. Habilita os botões dos jogadores na sua rodada e desabilita os botões do adversário.

Parameters:

`event` - (ActionEvent) botão escolhido pelo jogador.

vencedorRodadaC1

```
private Jogador vencedorRodadaC1()
```

Verifica Qual o jogador vencedor, caso a característica escolhida tenha sido a 1.

Verifica o tipo do jogo, para chamar o método da classe correta para comparação.

Returns:

(Jogador) jogador vencedor.

vencedorRodadaC2

```
private Jogador vencedorRodadaC2()
```

Verifica Qual o jogador vencedor, caso a característica escolhida tenha sido a 2.

Verifica o tipo do jogo, para chamar o método da classe correta para comparação.

Returns:
(Jogador) jogador vencedor.

vencedorRodadaC3

```
private Jogador vencedorRodadaC3()
```

Verifica Qual o jogador vencedor, caso a característica escolhida tenha sido a 3.

Verifica o tipo do jogo, para chamar o método da classe correta para comparação.

Returns:
(Jogador) jogador vencedor.

vencedorRodadaC4

```
private Jogador vencedorRodadaC4()
```

Verifica Qual o jogador vencedor, caso a característica escolhida tenha sido a 4.

Verifica o tipo do jogo, para chamar o método da classe correta para comparação.

Returns:
(Jogador) jogador vencedor.

vencedorRodadaC5

```
private Jogador vencedorRodadaC5()
```

Verifica Qual o jogador vencedor, caso a característica escolhida tenha sido a 5.

Verifica o tipo do jogo, para chamar o método da classe correta para comparação.

Returns:
(Jogador) jogador vencedor.

verificaVencedor

```
private void verificaVencedor(javafx.event.ActionEvent event,
                              Jogador vencedor)
```

Método responsável por verificar o vencedor da rodada.

Caso o jogo tenha finalizado (algum jogador ficou sem cartas), chama a tela do vencedor.

Caso haja empate, coloca uma mensagem na tela.

Parameters:

`event` - (ActionEvent) será passado no DBUtils.

`vencedor` - (Jogador) aquele que venceu a partida.

Module com.example.supertrunfo
Package poo.trabalhofinal.supertrunfo.gui.controllers

Class LoginContoller

java.lang.Object
    poo.trabalhofinal.supertrunfo.gui.controllers.LoginContoller

All Implemented Interfaces:

`javafx.fxml.Initializable`

```
public class LoginContoller
extends Object
implements javafx.fxml.Initializable
```

Classe LoginContoller

Classe responsável por intermediar a relação entre a interface (GUI) da *Tela de login* e o programa.

Classe responsável por logar o jogador para que ele possa jogar e seus dados serem alterados quando necessário.

Recebe as solicitações da interface e trata os eventos de acordo com o esperado no programa.

Implementa a interface *Initializable* do JavaFX, que define a assinatura do método de inicialização de um controller da tela.

Field Summary

Fields

Modifier and Type	Field	Description
`javafx.scene.control.Label`	`alerta1`	Elemento FXML que coloca uma mensagem de erro na tela (para 'login' do usuário 1), caso haja problema para logar.
`javafx.scene.control.Label`	`alerta2`	Elemento FXML que coloca uma mensagem de erro na tela (para 'login' do usuário 2), caso haja problema para logar.
`javafx.scene.control.Button`	`cadastrese1`	Elemento FXML (botão) que ao ser clicado leva à tela de cadastro de usuário (fica na parte de 'login' de primeiro usuário).
`javafx.scene.control.Button`	`cadastrese2`	Elemento FXML (botão) que ao ser clicado leva à tela de cadastro de usuário (fica na parte de 'login' de segundo usuário).

private Jogador<?>	jogador1	(Jogador A) atributo que representa o primeiro jogador logado.
private Jogador<?>	jogador2	(Jogador B) atributo que representa o segundo jogador logado.
private boolean	jogadorLogado1	(boolean) que representa se o primeiro jogador foi logado.
private boolean	jogadorLogado2	(boolean) que representa se o segundo jogador foi logado.
javafx.scene.control.Button	login1	Elemento FXML (botão) que, ao ser clicado, permite que o primeiro usuário faça 'login' (se os dados estiverem corretos).
javafx.scene.control.Button	login2	Elemento FXML (botão) que, ao ser clicado, permite que o segundo usuário faça 'login' (se os dados estiverem corretos).
javafx.scene.control.TextField	nome1	Elemento FXML que pega o nome do primeiro usuário por meio da GUI.
javafx.scene.control.TextField	nome2	Elemento FXML que pega o nome do segundo usuário por meio da GUI.
javafx.scene.control.PasswordFi	senha1	Elemento FXML que pega a senha do primeiro usuário por meio da GUI.
javafx.scene.control.PasswordFi	senha2	Elemento FXML que pega a senha do segundo usuário por meio da GUI.
private String	tipo	('String') que representa o tipo de jogo (baralho).

Constructor Summary

Constructors

Constructor	Description
LoginContoller()	

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
void	initialize(URL url, ResourceBundle resourceBundle)	Método sobrescrito oriundo da interface *Initializable*.

private void	**irParaJogo**(javafx.event.ActionEvent event)	Método que dreciona os jogadores para o jogo.
void	**setTipo**(**String** tipo)	Método público que modifica o valor do atributo privado tipo, que representa o tipo do jogo.
private void	**validaJogador1**()	Método que faz validação de jogador.
private void	**validaJogador2**()	Método que faz validação de jogador.
private void	**validaPreenchidoJogador1**()	Verifica se o jogador 1 preencheu todos os campos.
private void	**validaPreenchidoJogador2**()	Verifica se o jogador 2 preencheu todos os campos.

Methods inherited from class java.lang.Object

clone , equals , finalize , getClass , hashCode , notify , notifyAll , toString , wait , wait , wait

Field Details

nome1

```
public javafx.scene.control.TextField nome1
```

Elemento FXML que pega o nome do primeiro usuário por meio da GUI.

nome2

```
public javafx.scene.control.TextField nome2
```

Elemento FXML que pega o nome do segundo usuário por meio da GUI.

senha1

```
public javafx.scene.control.PasswordField senha1
```

Elemento FXML que pega a senha do primeiro usuário por meio da GUI.

senha2

```
public javafx.scene.control.PasswordField senha2
```

Elemento FXML que pega a senha do segundo usuário por meio da GUI.

login1

```
public javafx.scene.control.Button login1
```

Elemento FXML (botão) que, ao ser clicado, permite que o primeiro usuário faça 'login' (se os dados estiverem corretos).

login2

```
public javafx.scene.control.Button login2
```

Elemento FXML (botão) que, ao ser clicado, permite que o segundo usuário faça 'login' (se os dados estiverem corretos).

alerta1

```
public javafx.scene.control.Label alerta1
```

Elemento FXML que coloca uma mensagem de erro na tela (para 'login' do usuário 1), caso haja problema para logar. Mostra que o usuário conseguiu logar também.

alerta2

```
public javafx.scene.control.Label alerta2
```

Elemento FXML que coloca uma mensagem de erro na tela (para 'login' do usuário 2), caso haja problema para logar. Mostra que o usuário conseguiu logar também.

cadastrese1

```
public javafx.scene.control.Button cadastrese1
```

Elemento FXML (botão) que ao ser clicado leva à tela de cadastro de usuário (fica na parte de 'login' de primeiro usuário).

cadastrese2

```
public javafx.scene.control.Button cadastrese2
```

Elemento FXML (botão) que ao ser clicado leva à tela de cadastro de usuário (fica na parte de 'login' de segundo usuário).

tipo

```
private String  tipo
```

('String') que representa o tipo de jogo (baralho).

jogadorLogado1

```
private boolean jogadorLogado1
```

(boolean) que representa se o primeiro jogador foi logado.

jogadorLogado2

```
private boolean jogadorLogado2
```

(boolean) que representa se o segundo jogador foi logado.

jogador1

```
private Jogador<?> jogador1
```

(Jogador <?>) atributo que representa o primeiro jogador logado.

jogador2

```
private Jogador<?> jogador2
```

(Jogador <?>) atributo que representa o segundo jogador logado.

Constructor Details

LoginContoller

```
public LoginContoller()
```

Method Details

setTipo

```
public void setTipo(String  tipo)
```

Método público que modifica o valor do atributo privado tipo, que representa o tipo do jogo.

Parameters:

`tipo` - ('String') tipo do jogo.

initialize

```
public void initialize(URL  url,
                       ResourceBundle  resourceBundle)
```

Método sobrescrito oriundo da interface *Initializable*.

Analiza o tipo do jogo e inicializa (instancia) a classe JogadoresRepositoryImpl de acordo com o tipo do jogo. Assim, por métodos dessa classe, faz conexão com o banco de dados e tenta logar os jogadores (alerta na tela se der erro). Se der certo o alerta mostra na tela que logou.

Se clicar no botão de login 1, tenta logar o primeiro usuário.

Se clicar em login 2, tenta logar o segundo usuário.

Se clicar em qualquer botão de cadastre-se leva o usuário à tela de cadastro.

Specified by:

`initialize` in interface `javafx.fxml.Initializable`

Parameters:

`url` - (URL) do elemento fxml que está sendo carregado.

`resourceBundle` - (ResourceBundle) é fornecido como convenção para permitir o acesso a recursos adicionais.

validaJogador1

```
private void validaJogador1()
                    throws InformacaoInvalidaException
```

Método que faz validação de jogador.

Verifica se o jogador 1 já está logado e se o nome inserido para o jogador 2 é igual o do jogador 1. Impede que o usuário logue e jogue contra si mesmo.

Throws:

`InformacaoInvalidaException` - Se o mesmo jogador tentar logar em ambas as telas.

validaJogador2

```
private void validaJogador2()
                    throws InformacaoInvalidaException
```

Método que faz validação de jogador.

Verifica se o jogador 2 já está logado e se o nome inserido para o jogador 1 é igual o do jogador 2. Impede que o usuário logue e jogue contra si mesmo.

Throws:

`InformacaoInvalidaException` - Se o mesmo jogador tentar logar em ambas as telas.

irParaJogo

```
private void irParaJogo(javafx.event.ActionEvent event)
```

Método que dreciona os jogadores para o jogo. Muda a tela para tela do jogo.

Coloca uma mensagem de alerta caso haja algum erro ao tentar conectar ao jogo.

Parameters:

`event` - (ActionEvent) evento que será enviado para tela de jogo pelo change scene.

validaPreenchidoJogador1

```
private void validaPreenchidoJogador1()
                              throws InformacaoInvalidaException
```

Verifica se o jogador 1 preencheu todos os campos.

Throws:

`InformacaoInvalidaException` - Caso um dos campos não tenha sido preenchidos ao clicar em logar.

validaPreenchidoJogador2

```
private void validaPreenchidoJogador2()
                              throws InformacaoInvalidaException
```

Verifica se o jogador 2 preencheu todos os campos.

Throws:

`InformacaoInvalidaException` - Caso um dos campos não tenha sido preenchidos ao clicar em logar.

Module com.example.supertrunfo
Package poo.trabalhofinal.supertrunfo.gui.controllers

Class MenuController

java.lang.Object
    poo.trabalhofinal.supertrunfo.gui.controllers.MenuController

All Implemented Interfaces:

`javafx.fxml.Initializable`

```
public class MenuController
extends Object
implements javafx.fxml.Initializable
```

Classe MenuController

Classe responsável por intermediar a relação entre a interface (GUI) da *Tela de menu* e o programa.

A tela de menu mostra as ações que o usuário pode fazer dentro do FuJu trunfo.

Recebe as solicitações da interface e trata os eventos de acordo com o esperado no programa.

Implementa a interface *Initialize* do JavaFX, que define a assinatura do método de inicialização de um controller da tela.

Field Summary

Fields

Modifier and Type	Field	Description
`javafx.scene.control.Label`	`alerta`	Elemento FXML que coloca uma mensagem de alerta (erro) na tela caso algo não aconteça como esperado.
`javafx.scene.control.Button`	`jogar`	Elemento FXML botão que ao ser clicado leva a uma tela de opção de baralho para direcionar o jogado ao jogo.
`javafx.scene.control.Button`	`novaCarta`	Elemento FXML botão que ao ser clicado leva a uma tela de cadastro de nova carta.
`javafx.scene.control.Button`	`novoJogador`	Elemento FXML botão que ao ser clicado leva a uma tela de cadastro de novo jogador.
`javafx.scene.control.Button`	`regras`	Elemento FXML botão que ao ser clicado abre um link com um pdf com instruções e regras do jogo.
`javafx.scene.control.Button`	`verCartas`	Elemento FXML botão que ao ser clicado leva a uma tela na qual o usuário pode ver as

cartas existentes no jogo (todos os baralhos).

Constructor Summary

Constructors

Constructor	Description
`MenuController()`	

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
private void	`abrirPDF()`	Método responsável por direcionar o usuário para um link.
void	`initialize(URL url, ResourceBundle resourceBundle)`	Realiza as ações conforme o botão escolhido pelo usuário.

Methods inherited from class java.lang.Object

`clone` , `equals` , `finalize` , `getClass` , `hashCode` , `notify` , `notifyAll` , `toString` , `wait` , `wait` , `wait`

Field Details

jogar

```
public javafx.scene.control.Button jogar
```

Elemento FXML botão que ao ser clicado leva a uma tela de opção de baralho para direcionar o jogado ao jogo.

novoJogador

```
public javafx.scene.control.Button novoJogador
```

Elemento FXML botão que ao ser clicado leva a uma tela de cadastro de novo jogador.

novaCarta

```
public javafx.scene.control.Button novaCarta
```

Elemento FXML botão que ao ser clicado leva a uma tela de cadastro de nova carta.

verCartas

```
public javafx.scene.control.Button verCartas
```

Elemento FXML botão que ao ser clicado leva a uma tela na qual o usuário pode ver as cartas existentes no jogo (todos os baralhos).

regras

```
public javafx.scene.control.Button regras
```

Elemento FXML botão que ao ser clicado abre um link com um pdf com instruções e regras do jogo.

alerta

```
public javafx.scene.control.Label alerta
```

Elemento FXML que coloca uma mensagem de alerta (erro) na tela caso algo não aconteça como esperado.

Constructor Details

MenuController

```
public MenuController()
```

Method Details

initialize

```
public void initialize(URL  url,
                       ResourceBundle  resourceBundle)
```

Realiza as ações conforme o botão escolhido pelo usuário.

Caso a escolha seja jogar ou cadastrar cartas, chama a tela de opção. A diferença é que o primeiro envia o campo cadastro como false e o segundo como true. Assim, é possível diferenciar qual botão levou à tela de opção e realizar as ações corretas.

Se o usuário escolher o botão de cadastro de usuário, ele é direcionado para a tela de cadastro de jogador.

Se o usuário escolher o botão de ver cartas, ele pe direcionado para tela de ver cartas.

Specified by:

`initialize` in interface `javafx.fxml.Initializable`

Parameters:

`url` - (URL) do elemento fxml que está sendo carregado.

`resourceBundle` - (ResourceBundle) é fornecido como convenção para permitir o acesso a recursos adicionais.

abrirPDF

```
private void abrirPDF()
```

Método responsável por direcionar o usuário para um link. Verifica se o ambiente desktop atual suporta a classe Desktop, que faz operações de integração com a área de trabalho do ambiente. Caso não suporte, manda um alerta na tela. Caso seja Desktop Supported, ele tenta abrir o link da url. Se der erro mostra uma mensagem de alerta.

Module com.example.supertrunfo
Package poo.trabalhofinal.supertrunfo.gui.controllers

Class OpcaoController

java.lang.Object
poo.trabalhofinal.supertrunfo.gui.controllers.OpcaoController

All Implemented Interfaces:

`javafx.fxml.Initializable`

```
public class OpcaoController
extends Object
implements javafx.fxml.Initializable
```

Classe OpcaoController

Classe responsável por intermediar a relação entre a interface (GUI) da *Tela de opção* e o programa.

A tela de opção mostra as opções de jogo/baralho disponíveis para o jogador escolher ao cadastrar cartas ou jogar.

Recebe as solicitações da interface e trata os eventos de acordo com o esperado no programa.

Implementa a interface *Initialize* do JavaFX, que define a assinatura do método de inicialização de um controller da tela.

Field Summary

Fields

Modifier and Type	Field	Description
`private boolean`	`cadastro`	Variável booleana para indicar se essa tela foi chamada após o usuário escolher cadastrar carta ou jogar o jogo.
`javafx.scene.control.Button`	`gatos`	Elemento FXML Button que indica que o usuário escolheu gatos.
`javafx.scene.control.Button`	`linguagemProg`	Elemento FXML Button que indica que o usuário escolheu linguagens de programação.
`javafx.scene.control.Button`	`personagem`	Elemento FXML Button que indica que o usuário escolheu personagem.

Constructor Summary

Constructors

Constructor	Description
`OpcaoController()`	

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
void	`initialize(URL url, ResourceBundle resourceBundle)`	Método sobrescrito oriundo da interface *Initializable*.
void	`setCadastro(boolean cadastro)`	Método público que permite modificar o valor da variável cadastro (atributo privado).

Methods inherited from class java.lang.Object

`clone` , `equals` , `finalize` , `getClass` , `hashCode` , `notify` , `notifyAll` , `toString` , `wait` , `wait` , `wait`

Field Details

personagem

```
public javafx.scene.control.Button personagem
```

Elemento FXML Button que indica que o usuário escolheu personagem.

gatos

```
public javafx.scene.control.Button gatos
```

Elemento FXML Button que indica que o usuário escolheu gatos.

linguagemProg

```
public javafx.scene.control.Button linguagemProg
```

Elemento FXML Button que indica que o usuário escolheu linguagens de programação.

cadastro

```
private boolean cadastro
```

Variável booleana para indicar se essa tela foi chamada após o usuário escolher cadastrar carta ou jogar o jogo.

Constructor Details

OpcaoController

```
public OpcaoController()
```

Method Details

setCadastro

```
public void setCadastro(boolean cadastro)
```

Método público que permite modificar o valor da variável cadastro (atributo privado).

Parameters:

`cadastro` - (boolean) novo valor associado à variável cadastro.

initialize

```
public void initialize(URL  url,
                       ResourceBundle  resourceBundle)
```

Método sobrescrito oriundo da interface *Initializable*.

Define os venetos que serão tratados ao usuário clicar em um dos botões.

Chama uma nova tela e passa a escolha do usuário (personagem, gatos, linguagem de programação).

Caso o cadastro seja true, chama uma tela para cadastrar nova carta, se for false, quer dizer que a tela opção apareceu ao usuário escolher a opção jogar, mudando para tela de login.

Specified by:

`initialize` in interface `javafx.fxml.Initializable`

Parameters:

`url` - (URL) do elemento fxml que está sendo carregado.

`resourceBundle` - (ResourceBundle) é fornecido como convenção para permitir o acesso a recursos adicionais.

Module com.example.supertrunfo
Package poo.trabalhofinal.supertrunfo.gui.controllers

Class TelaInicialController

java.lang.Object
    poo.trabalhofinal.supertrunfo.gui.controllers.TelaInicialController

All Implemented Interfaces:

`javafx.fxml.Initializable`

```
public class TelaInicialController
extends Object
implements javafx.fxml.Initializable
```

Classe TelaInicialController

Classe responsável por intermediar a relação entre a interface (GUI) da *Tela Inicial* e o programa.

Recebe as solicitações da interface e trata os eventos de acordo com o esperado no programa.

Implementa a interface *Initializable* do JavaFX, que define a assinatura do método de inicialização de um controller da tela.

Field Summary

Fields

Modifier and Type	Field	Description
`javafx.scene.layout.AnchorPane`	`pane`	AnchorPane (Layout/Contêiner responsável pela posição dos elementos de forma absoluta dentro dele) pane: elemento no qual se encontram os elementos da interface na Tela Inicial -> uado para capturar o mouse event.

Constructor Summary

Constructors

Constructor	Description
`TelaInicialController()`	

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
void	initialize(URL location, ResourceBundle resources)	Método sobrescrito oriundo da interface *Initializable*.

Methods inherited from class java.lang.Object

clone , equals , finalize , getClass , hashCode , notify , notifyAll , toString , wait , wait , wait

Field Details

pane

```
public javafx.scene.layout.AnchorPane pane
```

AnchorPane (Layout/Contêiner responsável pela posição dos elementos de forma absoluta dentro dele) pane: elemento no qual se encontram os elementos da interface na Tela Inicial -> uado para capturar o mouse event.

Constructor Details

TelaInicialController

```
public TelaInicialController()
```

Method Details

initialize

```
public void initialize(URL  location,
                       ResourceBundle  resources)
```

Método sobrescrito oriundo da interface *Initializable*.

Define que ao clicar com mouse na tela (AnchorPane) o evento seja capturado e a tela mude para a tela de menu.

Specified by:

`initialize` in interface `javafx.fxml.Initializable`

Parameters:

`location` - (URL) do elemento fxml que está sendo carregado.

`resources` - (ResourceBundle) é fornecido como convenção para permitir o acesso a recursos adicionais.

Module com.example.supertrunfo
Package poo.trabalhofinal.supertrunfo.gui.controllers

Class VencedorController

java.lang.Object
    poo.trabalhofinal.supertrunfo.gui.controllers.VencedorController

---

```
public class VencedorController
extends Object
```

Classe VencedorController

Classe responsável por intermediar a relação entre a interface (GUI) da *Tela de vencedor* e o programa.

Recebe as solicitações da interface e trata os eventos de acordo com o esperado no programa. Aqui, mostra o nome do jogador vencedor e atualiza os dados (pontuação) de ambos os jogadores.

Implementa a interface *Initializable* do JavaFX, que define a assinatura do método de inicialização de um controller da tela.

Field Summary

Fields

Modifier and Type	Field	Description
`private final boolean`	`empate`	Atributo do tipo booleano que indica se houve um empate na partida.
`private final JogadoresRepository<?>`	`jogadoresRepository`	Atributo privado que conecta ao banco de dados de jogadores e atualiza as pontuações após o jogo.
`private final Jogador`	`jogadorFracasado`	Atributo do tipo Jogador que representa o perdedor da partida.
`private final Jogador`	`jogadorVencedor`	Atributo do tipo Jogador que representa o vencedor da partida.
`private javafx.scene.control.Button`	`sair`	Elemento FXML que representa um botão, que ao ser clicado, permite ao usuário sair da tela de vencedor.
`private javafx.scene.control.Label`	`vencedor`	Elemento FXML Label, que mostra o nome do vencedor na tela.

Constructor Summary

Constructors

Constructor	Description
`VencedorController()`	

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
`void`	`initialize()`	Método sobrescrito oriundo da interface *Initializable*.

Methods inherited from class java.lang.Object

`clone` , `equals` , `finalize` , `getClass` , `hashCode` , `notify` , `notifyAll` , `toString` , `wait` , `wait` , `wait`

Field Details

sair

```
private javafx.scene.control.Button sair
```

Elemento FXML que representa um botão, que ao ser clicado, permite ao usuário sair da tela de vencedor.

vencedor

```
private javafx.scene.control.Label vencedor
```

Elemento FXML Label, que mostra o nome do vencedor na tela.

jogadoresRepository

```
private final JogadoresRepository<?> jogadoresRepository
```

Atributo privado que conecta ao banco de dados de jogadores e atualiza as pontuações após o jogo.

jogadorVencedor

```
private final Jogador jogadorVencedor
```

Atributo do tipo Jogador que representa o vencedor da partida.

jogadorFracasado

```
private final Jogador jogadorFracasado
```

Atributo do tipo Jogador que representa o perdedor da partida.

empate

```
private final boolean empate
```

Atributo do tipo booleano que indica se houve um empate na partida.

Constructor Details

VencedorController

```
public VencedorController()
```

Method Details

initialize

```
public void initialize()
```

Método sobrescrito oriundo da interface *Initializable*.

Faz a atualização da pontuação dos jogadores no banco de dados. Caso haja algum erro ao conectar no banco de dados coloca um alerta na tela.

Coloca na tela o nome do jogador vencedor.

Se o usuário apertar no botão 'sair', muda a tela para a tela de *menu*.

Module com.example.supertrunfo
Package poo.trabalhofinal.supertrunfo.gui.controllers

Class VerCartasController

java.lang.Object
    poo.trabalhofinal.supertrunfo.gui.controllers.VerCartasController

All Implemented Interfaces:

`javafx.fxml.Initializable`

```
public class VerCartasController
extends Object
implements javafx.fxml.Initializable
```

Classe VerCartasController

Classe responsável por intermediar a relação entre a interface (GUI) da *Tela para ver cartas* e o programa.

Mostra na tela todas as cartas do jogo, uma a uma.

Recebe as solicitações da interface e trata os eventos de acordo com o esperado no programa.

Implementa a interface *Initializable* do JavaFX, que define a assinatura do método de inicialização de um controller da tela.

Field Summary

Fields

Modifier and Type	Field	Description
`javafx.scene.control.Label`	`alerta`	Elemento FXML que coloca uma mensagem de erro na tela caso algo dê errado.
`javafx.scene.control.Button`	`anterior`	Elemento FXML que permite o usuário ver a carta anterior.
`javafx.scene.control.Label`	`c1`	Elemento FXML responsável por mostrar qual é a primeira característica da carta na tela.
`javafx.scene.control.Label`	`c2`	Elemento FXML responsável por mostrar qual é a segunda característica da carta na tela.
`javafx.scene.control.Label`	`c3`	Elemento FXML responsável por mostrar qual é a terceira característica da carta na tela.

javafx.scene.control.Label	**c4**	Elemento FXML responsável por mostrar qual é a quarta característica da carta na tela.
javafx.scene.control.Label	**c5**	Elemento FXML responsável por mostrar qual é a quinta característica da carta na tela.
javafx.scene.control.Label	**caracteristica1**	Elemento FXML responsável por mostrar o valor da primeira característica da carta na tela.
javafx.scene.control.Label	**caracteristica2**	Elemento FXML responsável por mostrar o valor da segunda característica da carta na tela.
javafx.scene.control.Label	**caracteristica3**	Elemento FXML responsável por mostrar o valor da terceira característica da carta na tela.
javafx.scene.control.Label	**caracteristica4**	Elemento FXML responsável por mostrar o valor da quarta característica da carta na tela.
javafx.scene.control.Label	**caracteristica5**	Elemento FXML responsável por mostrar o valor da quinta característica da carta na tela.
private **ArrayList** <**Carta**>	**cartas**	ArrayList que representa a lista de todas as cartas do jogo.
javafx.scene.control.Label	**classificacao**	Elemento FXML responsável por mostrar a classificação na tela.
javafx.scene.control.Label	**descricao3**	Elemento FXML que descreve o valor da característica 3.
javafx.scene.control.Label	**descricao4**	Elemento FXML que descreve o valor da característica 4.
javafx.scene.control.Label	**descricao5**	Elemento FXML que descreve o valor da característica 5.
private static int	**i**	Índice para ver uma carta da lista de cartas.
javafx.scene.image.ImageView	**imagem**	Elemento FXML responsável por mostrar uma imagem na tela.
javafx.scene.control.Label	**nome**	Elemento FXML responsável por mostrar o nome na tela.
javafx.scene.control.Button	**proximo**	Elemento FXML que permite o usuário ver a carta posterior.

javafx.scene.control.Button	sair	Elemento FXML que permite o usuário voltar ao menu.
javafx.scene.control.Label	tipo	Elemento FXML que mostra qual o tipo da carta.
javafx.scene.image.ImageView	trunfo	Elemento FXML responsável por colocar a imagem de FuJu Trunfo, caso a carta seja um supertrunfo.

Constructor Summary

Constructors

Constructor	Description
VerCartasController()	

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	initialize(URL url, ResourceBundle resourceBundle)	Método sobrescrito oriundo da interface *Initializable*.
private void	mostrarCartas(Carta carta)	Imprime na tela da GUI (por meio dos Labels) os dados de cada carta.

Methods inherited from class java.lang.Object

clone , equals , finalize , getClass , hashCode , notify , notifyAll , toString , wait , wait , wait

Field Details

imagem

```
public javafx.scene.image.ImageView imagem
```

Elemento FXML responsável por mostrar uma imagem na tela.

trunfo

```
public javafx.scene.image.ImageView trunfo
```

Elemento FXML responsável por colocar a imagem de FuJu Trunfo, caso a carta seja um supertrunfo.

classificacao

```
public javafx.scene.control.Label classificacao
```

Elemento FXML responsável por mostrar a classificação na tela.

nome

```
public javafx.scene.control.Label nome
```

Elemento FXML responsável por mostrar o nome na tela.

caracteristica1

```
public javafx.scene.control.Label caracteristica1
```

Elemento FXML responsável por mostrar o valor da primeira característica da carta na tela.

caracteristica2

```
public javafx.scene.control.Label caracteristica2
```

Elemento FXML responsável por mostrar o valor da segunda característica da carta na tela.

caracteristica3

```
public javafx.scene.control.Label caracteristica3
```

Elemento FXML responsável por mostrar o valor da terceira característica da carta na tela.

caracteristica4

```
public javafx.scene.control.Label caracteristica4
```

Elemento FXML responsável por mostrar o valor da quarta característica da carta na tela.

caracteristica5

```
public javafx.scene.control.Label caracteristica5
```

Elemento FXML responsável por mostrar o valor da quinta característica da carta na tela.

c1

```
public javafx.scene.control.Label c1
```

Elemento FXML responsável por mostrar qual é a primeira característica da carta na tela.

c2

```
public javafx.scene.control.Label c2
```

Elemento FXML responsável por mostrar qual é a segunda característica da carta na tela.

c3

```
public javafx.scene.control.Label c3
```

Elemento FXML responsável por mostrar qual é a terceira característica da carta na tela.

c4

```
public javafx.scene.control.Label c4
```

Elemento FXML responsável por mostrar qual é a quarta característica da carta na tela.

c5

```
public javafx.scene.control.Label c5
```

Elemento FXML responsável por mostrar qual é a quinta característica da carta na tela.

descricao3

```
public javafx.scene.control.Label descricao3
```

Elemento FXML que descreve o valor da característica 3.

descricao4

```
public javafx.scene.control.Label descricao4
```

Elemento FXML que descreve o valor da característica 4.

descricao5

```
public javafx.scene.control.Label descricao5
```

Elemento FXML que descreve o valor da característica 5.

tipo

```
public javafx.scene.control.Label tipo
```

Elemento FXML que mostra qual o tipo da carta.

sair

```
public javafx.scene.control.Button sair
```

Elemento FXML que permite o usuário voltar ao menu.

anterior

```
public javafx.scene.control.Button anterior
```

Elemento FXML que permite o usuário ver a carta anterior.

proximo

```
public javafx.scene.control.Button proximo
```

Elemento FXML que permite o usuário ver a carta posterior.

alerta

```
public javafx.scene.control.Label alerta
```

Elemento FXML que coloca uma mensagem de erro na tela caso algo dê errado.

cartas

```
private ArrayList <Carta> cartas
```

ArrayList que representa a lista de todas as cartas do jogo.

i

```
private static int i
```

Índice para ver uma carta da lista de cartas.

Constructor Details

VerCartasController

```
public VerCartasController()
```

Method Details

initialize

```
public void initialize(URL  url,
                       ResourceBundle  resourceBundle)
```

Método sobrescrito oriundo da interface *Initializable*.

Tenta conectar com o banco de dados por meio de étodos da classe CartasRepository.

Caso o usuário aperte em um botão próximo, o contador é incrementado. Enquanto não tiver atingido o final da lista de cartas, a carta aparesentada na tela é mudada para a próxima carta da lista. Caso atinja seu fim, coloca um alerta na tela falando que chegou ao fim.

Se o usuário apertar no botão de anterior, o contado é decrementado até chegar ao início (primeiro elemento) da lista.

AO cilcar no botão sair, ele retorna para tela de menu.

Specified by:

`initialize` in interface `javafx.fxml.Initializable`

Parameters:

`url` - (URL) do elemento fxml que está sendo carregado.

`resourceBundle` - (ResourceBundle) é fornecido como convenção para permitir o acesso a recursos adicionais.

mostrarCartas

```
private void mostrarCartas(Carta carta)
```

Imprime na tela da GUI (por meio dos Labels) os dados de cada carta. Coloca a imagem da carta na tela.

Parameters:

`carta` - (Carta) carta que está sendo mostrada

UML DO PROJETO: