Classes, Methods and Variables Guide

Use it as a guide for APP – Acompanhamento de Projetos Parlamentares and Jogo do Morro

Version 1.0 (14/04/2014)

Made by: Eduardo de Oliveira Castro 12/0115921
Luciano Henrique Nunes de Almeida 11/0063392
Matheus Carneiro Godinho de Morais Sá 12/0018811
Simião Correia Lima de Carvalho 12/0022265

Summary

1. view

1.1. com.mds.app.view

Folder: View
Package: com.mds.app.view
Class: Busca.java

Package:
com.mds.app.view: Package containing all the views elements from the application

Imports:
android.app.Activity: A single thing that the user can do. It basically creates a window where you can place your UI with other elements.
android.app.ProgressDialog: A dialog showing a progress indicator and an optional text message or view.
android.content.Intent: Intent is a messaging object you can use to request an action from another app component. It facilitates the communication between components.
android.os.AsyncTask: Allows to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers.
android.os.Bundle: A mapping from String values to various Parcelable types.
android.os.StrictMode: It detects things you migh be doing by accident and brings them to your attention so you can fix it.
android.util.Log: Basically a Log.
android.view.Menu: Interface for managing the items in a menu.
android.view.View: Represents the basic building block for UI components.
android.view.View.OnClickListener: Interface definition for a callback to be invoked when a view is clicked.
android.widget.EditText: Basically a Text Field for entries.
android.widget.ImageButton: Basically a button with an image.
android.widget.Spinner: Basically a list of options.
android.widget.Toast: A view containing a quick little message for the user.

Class:
Busca: Responsible for searches.
PesquisarProjetoTask: Inner class inside Busca class. It searches for some elements.

Variables:
Progress Dialog progressDialog: Variable that receives an object of ProgressDialog type responsible for handle a window informing about something. In this case the progress of the searching.
ImageButton botaoPesquisar: Variable from ImageButton type, it basically creates a button. If this button is pressed the software will follow the botaoPesquisar_addListener() method instructions.
BuscaController buscaController: Receives an object of BuscaController type. In this class it purporses is basically to prepare the search atributes for a proper search.
ConexaoInternet conexao: It checks the internet connection.
StrictMode.ThreadPolicy policy: StrictMode is a developer tool which detects things you might be doing by accident and brings them to your attention so you can fix it, usally is used to catch accidental disk or network access.
R.id.okbutton: The button that we defined at res folder.
Spinner spinner1: Basically a list where the user can choose an option.
Spinner spinner2: Basically a list where the user can choose an option.
EditText numeroTexto: A text field that receives the text number.
EditText anoTexto: A text field that receives the text year.
EditText dataInicialTexto: A text field that receives the text start date.

EditText nomeAutorTexto: A text field that receives the text author name.
Spinner siglaPartidoTexto: A text field that receives the policial party acronym.
List<ProjetoModel> listaProjetos: Receives a list from buscaController.procurar().
Intent intent: An intent is a messaging object that requests an action from another app component. Basically facilitates the communication between components. Here we use it to create a communication between Busca and Lista classes.

Methods:

conexao.ChecarConexaoInternet(): Only check the connection.
buscaController.setTemConexao(true): If there is a connection so it set it as true.
onCreate(Bundle savedInstanceState): default method from Android, similar to before method from JUnit it basically prepare the application. It's here where you initialize your activity.
setContentView(R.layout.activity_busca): Set the activity content to an explicit view, in this case the content was defined by the R.layout.activity_busca that was created by us and it is described at res/layout/activity_busca.xml).
botaoPesquisar_addListener(): adds a listener to the set of listeners that are sent events through the life of an animation, such as start, repeat and end. At this case this is an activity that starts a search routine when the search button is pressed. It asks for the text number, the text year, the author name and the political party name, validates it and call another method who will actually search and show the result.
botaoPesquisar.setOnClickListener(): Where the magic from the last method happens. Here is where something start happening after the user click.
onClick(View v): It is inside the previous method, botaoPesquisar.setOnClickListener(), where we start initializing some variables and objects and now onClick() defines what will happen when the button is pressed. Basically this is the part of the code that validates the entries and call the PesquisarProjetoTask().execute() that really makes the search or return a message saying that the entries are wrong.
onCreateOptionsMenu(Menu menu): Inflate the menu, this adds items to the action bar if it is present.
onPreExecute(): Do something before executing the flow. Basically shows a dialog saying that the software is receiving data and searching.
doInBackground(Void... params): Basically store the result from the search(buscaController.procurar()) into a list variable(listaProjetos).
onPostExecute(final List<ProjetoModel> result): Part of code executed after the search. It dismiss the progressDialog and send the result list to ListaController controller. It also creates a new Intent with Busca and Lista classes and starts an Activity using it.

Folder: View
Package: com.mds.app.view
Class: Lista.java

Package:

com.mds.app.view: Package containing all the views elements from the application.

Imports:

android.app.Activity: A single thing that the user can do. It basically creates a window where you can place your UI with other elements.

android.content.Intent: Intent is a messaging object you can use to request an action from another app component. It facilitates the communication between components.

android.os.Bundle: A mapping from String values to various Parcelable types.

android.util.Log: Basically a Log.

android.view.Menu: Interface for managing the items in a menu.

android.view.View: Represents the basic building block for UI components.

android.widget.AdapterView: Is a view whose children are determined by an Adapter.

android.widget.ListView: Shows items in a vertically scrolling list.

android.widget.Toast: A view containing a quick little message for the user.

com.mds.app.R: R class from Android. Auto-generated and NEVER modify anything from this class.

com.mds.app.controller.ListaController: Controller responsible to generate and return a list of projects or auto-complete informations about any project.

com.mds.app.util.StableArrayAdapter: This class is responsible to prepare and generate a list of elements to any activity or view.

Class:

Lista: Shows a list of projects or information about any project.

Variables:

ListaController listaController: An instance from ListaController object.

ArrayList<String> stringProjetos: Receives informations about all the projects managed by ListaController. It receives informations like name, number and author.

Bundle savedInstanceState: A bundle is a mapping from String values, in this case saving some instance states.

ListView listView: Basically a list of elements.

StableArrayAdapter adapter: It prepares the elements into a list to be shown by listView.

android.R.layout.simple_list_item_1: An list layout pre-determined by Android library. It is not defined into our layout folder as the layouts created by us.

AdapterView<?> parent: Represents the AdapterView that it is the parent from this one(that it is showing the click/activity)

View view: Just a view.

int position: Represents the position from any project.

long id: Not used.

Intent i: Places the communication between Lista and Perfil classes for a new activity

Menu menu: Basically a menu

R.menu.main: Definition of the menu

message: Any message to be shown

Toast.LENGTH_LONG: The length from the message

CharSequence message: The message that should be shown

Methods:

onCreate(Bundle savedInstanceState): default method from Android, similar to before method from JUnit it basically prepare the application. It's here where you initialize your activity.

Log.i("LISTA", stringProjetos.toString()): Basically a log function. In this case it sends the list with the projects to the log file.

onItemClick(AdapterView<?> parent, View view, int position, long id): When this item is clicked we start this flow.

run(): Nothing special, just start the flow defined inside the onItemClick() and is shown when the user clicks.

onCreateOptionsMenu(Menu menu): Inflate the menu, this adds items to the action bar if it is present.

longToast(CharSequence message): It shows the Toast message.

Folder: View
Package: com.mds.app.view
Class: MenuPrincipal.java

Packages:

com.mds.app.view: Package containing all the views elements from the application.

Imports:

android.app.Activity: A single thing that the user can do. It basically creates a window where you can place your UI with other elements.

android.content.Context: An interface to global information about an application environment. It allows access to application-specific resources and classes or launching activities, broadcasting and receiving intents, etc.

android.content.Intent: Intent is a messaging object you can use to request an action from another app component. It facilitates the communication between components.

android.os.Bundle: A mapping from String values to various Parcelable types.

android.util.Log: Basically a Log.

android.view.Menu: Interface for managing the items in a menu.

android.view.View.OnClickListener: Interface definition for a callback to be invoked when a view is clicked.

android.widget.ImageButton: Basically a button with an image.

com.mds.app.R: R class from Android. Auto-generated and NEVER modify anything from this class.

com.mds.app.controller.FavoritosController:

com.mds.app.controller.HistoricoController:

com.mds.app.controller.ListaController

com.mds.app.persistencia.Persistencia:

Class:

MenuPrincipal: A class for the main menu from the application.

Variables:

Context context: Represents the context of this class(in this case, the MenuPrincipal.class)

ImageButton busca: Just a button, in this case responsible for starting any search.

ImageButton sobre: Just a button, in this case responsible to open the about page.

ImageButton favoritos: Just a button, in this case responsible to open the favorites page.

ImageButton historico: Just a button, in this case responsible to open the historic page.

Bundle savedInstanceState: A bundle is a mapping from String values, in this case saving some instance states.

Persistencia persitencia: Responsible for the persistence from the application, it this class whom manages the data files.

String conteudoHistorico: Receives the historic files called from the persistence file.

String conteudoFavoritos: Receives the favorite files called from the persistence file.

Persistencia.getFileNameHistorico(): The name from the file name at the historic data file.

FavoritosController favoritosController: An instance from the FavoritosController class. We use it to call the popularProjetos() method.

HistoricoController historicoController: An instance from the HistoricoController class. We use it to call the populateProject() method.

Menu menu: Basically a menu that we defined in the res folder.

R.menu.main: The xml file that we create to define the menu.

R.id.imgbutton_busca: The file that we create to define the search button.

View v: Just a view.

Intent i: Places the communication between MenuPrincipal and Busca classes for a new activity.

Persistencia.getFileNameFavoritos(): Returns the file name at the favorited list.

FavoritosController.getProjetosFavoritados(): Returns the favorited projects.

String strConteudoFavoritos: The variable that receives the file name at the favorited list.

String strConteudoHistorico: The variable name that receives the historic content.

Methods:

onCreate(Bundle savedInstanceState): default method from Android, similar to before method from JUnit it basically prepare the application. It's here where you initialize your activity.

onCreateOptionsMenu(Menu menu): Inflate the menu, this adds items to the action bar if it is present.

busca_addListener(): adds a listener to the set of listeners that are sent events through the life of an animation, such as start, repeat and end. At this case this is an activity

onClick(View v):

sobre_addListener(): adds a listener to the set of listeners that are sent events through the life of an animation, such as start, repeat and end. At this case this is an activity that calls the Busca class and starts it flows.

favoritos_addListener(): adds a listener to the set of listeners that are sent events through the life of an animation, such as start, repeat and end. At this case this is an activity that calls the Favoritos class and starts it flows.

historico_addListener(): adds a listener to the set of listeners that are sent events through the life of an animation, such as start, repeat and end. At this case this is an activity that calls Historico class and starts it flows.

Folder: View
Package: com.mds.app.view
Class: Perfil.java

Packages:

com.mds.app.view: Package containing all the views elements from the application.

Imports:

android.app.Activity: A single thing that the user can do. It basically creates a window where you can place your UI with other elements.

android.content.Context:

android.content.Intent: Intent is a messaging object you can use to request an action from another app component. It facilitates the communication between components.

android.os.Bundle: A mapping from String values to various Parcelable types.

android.util.Log: Basically a Log.

android.view.Menu: Interface for managing the items in a menu.

android.view.View: Represents the basic building block for UI components.

android.view.View.OnClickListener: Interface definition for a callback to be invoked when a view is clicked.

android.widget.ImageButton: Basically a button with an image.

android.widget.TextView: Basically a view with a pre-defined text. Sometimes allow users to edit it.

android.widget.Toast: A view containing a quick little message for the user.

com.facebook.FacebookOPerationCanceledException: An exception to be throw in case of error canceling an operation.

com.facebook.Session: Controls the session from the communitacion between facebook and application.

com.facebook.SessionState: Controlls the state from the session.

com.facebook.UiLifecycleHelper : It automatically opens and save and restore the Active Session in a way that is similar to Android UI lifecycles.

com.facebook.widget.FacebookDialog: A set of methods for placing dialogs between facebook and the application/user.

com.mds.app.R: R class from Android. Auto-generated and NEVER modify anything from this class.

com.mds.app.controller.FavoritosController: Responsible for managing the list of favorited projects.

com.mds.app.controller.HistoricoController: Responsible for managing a historic of projects.

com.mds.app.controller.ListaController: Responsible for managing a list of projects.

com.mds.app.model.ProjetoModel: Defines the model from a project.

Variables:

ListaController listaController: Variable that will receive the instance of ListaController class.

ProjetoModel projetoAtual: Variable that will receive the actual project from ListaController class.

String stringProjetoCompleto: Variable that will receive the complete project name from listaController variable.

TextView text1: Only a text view that sometimes the user can edit. In this case it will receive the name of a project and follows the text type design defined by R.id.textoTipoProjeto.

TextView text2: Only a text view that sometimes the user can edit. In this case it will receive the number, year, abreviation and presentation date from a project and follows the text design defined by R.id.textoCarcTeristicasProjeto.

TextView text3: Only a text view that sometimes the user can edit. In this case it will receive the description from a project and follows the text design defined by R.id.textoDescricao.

TextView text4: Only a text view that sometimes the user can edit. In this case it will receive the parlamentar word and follows the text design defined by R.id.textoParlamentar.

TextView text5: Only a text view that sometimes the user can edit. In this case it will receive the parlamentar name, the political party name and its abreviation. It will follow the text design defined by R.id.textoCaracteristicasParlamentar.

TextView text6: Only a text view that sometimes the user can edit. In this case it will receive a link for more informations and follow the text design defined by R.id.textoMais.

TextView text7: Only a text view that sometimes the user can edit. In this case it will receive the status from the project and follows the text design defined by R.id.textoStatus.

ImageButton estrelaFavorito: An ImageButton from an star that says if a project it is favorited or not.

ImageButton botaoFacebook: A button for facebook sharing.

boolean favoritado: It says if the actual project is favorited or not.

Context context: Represents the context, in this case this class.

boolean isResumed: It says if the app is in use or paused. If it is in use the value is true and if it is paused the value is false.

UiLifecycleHelper uiHelper: It automatically opens and save and restore the Active Session in a way that is similar to Android UI lifecycles.

Session session: The session passed as parameter.

SessionState state: The state from the session passed as parameter.

Exception exception: An exception.

Bundle savedInstanceState: A bundle is a mapping from String values, in this case saving some instance states.

R.layout.activity_perfil: Activity Layout created at the layout folder for the Perfil class.

R.id.textoTipoProjeto: The textview created to receive the type of the project.

R.id.textoCaracteristicasProjeto: The textview created to receive the caracteristics from the project.

projetoAtual.getNumero(): It represents the number from the project.

projetoAtual.getAno(): It represents the year from the project.

projetoAtual.getSigla(): It represents the abreviation from the political party from the project.

projetoAtual.getData(): It represents the date from the project.

R.id.textoDescricao: The textview created to receive the project description.

projetoAtual.getExplicacao(): It represents the explanation about the project.

R.id.textoParlamentar: The textview created to receive the parlamentar name from the project.

R.id.textoCarcteristicasParlamentar(): The textview created to receive some caracteristics from the parlamentar.

projetoAtual.getParlamentar().getNome(): It represents the name from the parlamentar.

projetoAtual.getParlamentar().getPartido().getSiglaPartido(): It represents the abreviation from the political party from the parlamentar.

R.id.textoMais: The textview created to show more information about the project.

projetoAtual.getId(): Represents the id from the project.

R.id.textoStatus: The textview created to show the project status.

projetoAtual.getStatus(): It represents the status from the project.

int projetosNoHistorico: The number of the projects at the historic.

int maxProjetos: The max number of projects allocated at the historic.

String stringProjetoParaHistorico: It receives all the informations about the project(name, number, year, abreviation, date, explanation, status, parlamentar, etc)

HistoricoController historicoController: An instance for HistoricoController class.
HistoricoController.getProjetoMaisVelho().getNumero(): The number from the oldest project.
projetoAtual.getNumero(): The number from the actual project.
int requestCode: Request code for ActivityResult.
int resultCode: Result code for ActivityResult.
Intent data: Data for this intent.
Menu menu: Just a menu.
R.menu.main: Menu defined at res folder.
Activity activity: An activity passed as parameter.
View v: Basically a view passed as parameter.
shareDialog.present(): the present shareDialog to be passed as parameter for uiHelper.
listaController.getStringCompletaParaArquivo(): It returns all the informations about a project for another variable.
R.drawable.favorito: Favorite drawing. Basically a star colored.
R.drawable.naofavorito: "Not favorite" drawing. Basically a star uncolored.
String stringProjetoParaFavorito: String that receives all the informations about a project.
Bundle outState: State that need to be saved.

Methods:
call(Session session, SessionState state, Exception exception): Provides asynchronous notification of Session state changes.
onCreate(Bundle savedInstanceState): default method from Android, similar to before method from JUnit it basically prepare the application.
facebook_addListener():
ListaController.getProjetoAtual(): It returns the actual project.
(TextView) findViewById(R.id.textoMais): It says that the variable that receive this method will follow the view defined by this parameter..
text7.setText("Status..."): Set a text at the view atributed to text7 variable.
favoritar_addListener(): Listener for the fav action. It basically saves the "fav state" from the project.
Log.i("LOGGER", "Removendo..."): Sends a new message to be added at the log file.
historicoController.adicionar(projetoAtual, stringProjetoParaHistorico): Adds a new project to the historic. The first parameter it is the instance and the second one all the informations about the project.
historicoController.remover(HistoricoController.getProjetoMaisVelho(), HistoricoController, getStringProjetoMaisVelho()): It deletes a project from the historic. The first parameter returns which project its the older one, the second it is the parameter and the last one all the informations about this project.
onActivityResult(int requestCode, int resultCode, Intent data): It places the log in with facebook.
onError(FacebookDialog.PendingCall pendingCall, Exception error, Bundle data): In case of error during the log in try it returns which error.
onComplete(FacebookDialog.PendingCall pendingCall, Bundle data): In case of success during the log in try it returns a success message.
onCreateOptionsMenu(Menu menu): Inflate the menu, this adds items to the action bar if it is present.
getMenuInflater().inflate(R.menu.main, menu): It places the new menu, passed as parameter, in first plan.
facebook_addListener(): The listener that defines the share on facebook function.
botaoFacebook.setOnClickListener(new OnclickListener() { …): If the button is clicked so the project page will be shared on facebook.

estrelaFavorito.setImageResource(): It places the favorite image that follows the “state” from favoritado variable. If it is true so we will call a image that is colored and sinalizes that the project it was favorited, it is isnt so we call another image sinalizing that the project it wasnt favorited.

estrelaFavorito.setOnClickListener(new OnClickListener() { ...): It defines the activity of favoriting or unfavoriting a project, adding(or removing) it to favoritosController, to the log and setting the new image.

onResume(): From Facebook SDK. It defines what should be done if the app is on resume state.

onSaveInstanceState(Bundle outState): From Facebook SDK. It saves an instance state.

OnPause(): From Facebook SDK. It defines what should be done if the app is paused.

onDestroy(): From Facebook SDK. It defines what should be done if the app is closed/destroyed.

onSessionStateChange(Session session, SessionState state, Exception exception): From Facebook SDK. It defines what should be done if the session state is changed.

Folder: View
Package: com.mds.app.view
Class: Sobre.java

Package:

com.mds.app.view: Package containing all the views elements from the application.

Imports:

android.app.Activity: A single thing that the user can do. It basically creates a window where you can place your UI with other elements.

android.os.Bundle: A mapping from String values to various Parcelable types.

android.view.Menu: Interface for managing the items in a menu.

android.widget.ImageButton: Basically a button with an image.

com.mds.app.R: R class from Android. Auto-generated and NEVER modify anything from this class.

Class:

Sobre: Activity from About Page

Variables:

ImageButton voltar: Variable from ImageButton type, it basically creates a button. Aparently it's not been used here.

Bundle savedInstanceState: A bundle is a mapping from String values, in this case saving some instance states.

R.layout.activity_sobre: The layout from this activity defined in res/layout/activity_sobre.xml

Menu menu: Basically a menu.

R.menu.sobre: The menu defined for this activity in res/layout/sobre.xml.

Methods:

onCreate(Bundle savedInstanceState): default method from Android, similar to before method from JUnit it basically prepare the application. It's here where you initialize your activity.

setContentView(R.layout.activity_sobre): Set the activity content to an explicit view, in this case the content was defined by the R.layout.activity_sobre that was created by us and it is described at res/layout/activity_sobre.xml).

getMenuInflater().inflate(R.menu.sobre, menu): Returns a menu inflater with this context. Basically instantiate menu XML files into Menu objects.

2. Src

2.1. com.mds.app.controller

Folder: Src
Package: com.mds.app.controller
Class: AlteraArquivos.java

Methods:

// Writes a project at the archive
- void adicionar(ProjetoModel projeto, String conteudo);

ProjetoModel projeto: Parlamentar project that stores the project data

String conteudo: Stores the project content in string format

// Removes a project from the archive
- void remover(ProjetoModel projeto, String conteudo);

ProjetoModel projeto: Parlamentar Project that stores the project data

String conteudo: Stores the project content in string format

// Transform all the content in string
- String projetosEmString();

String conteudoProjetosFavoritados: Stores the content from all the projects favorited or at the historic.

// Takes the archive content and transforms it in projects
- void popularProjetos(String conteudoArquivo);

String conteudoArquivo: string that stores all the content from the archive

// Fills the list to be shown
- void popularListaComProjetos();

Folder: Src
Package: com.mds.app.controller
Class: FavoritosController.java

Variables:

static ArrayList<ProjetoModel> projetosFavoritados: List with all the favorited projects.
static ArrayList<String> projetosFavoritadosCompletoStr: List with all the project data favorited in string format.
Persistencia persistencia: Class instanceInstancia that communicates with the archive.
ArrayList<String> splitParts: List that stores the projects in string format.
final int separadoresPorProjeto: Constant that represents how many separators a project has.
final int numeroDeProjetosNoArquivo: Counts how many projects there are in an archive.
int numeroDeSeparadores: Counts how many separators there are in an archive.
PartidoModel partido: Stores the data from a political party.
ParlamentarModel parlamentar: Stores the data from a parlamentar.
ProjetoModel projeto: Stores the data from a project.
String stringProjeto: Temporary variable that stores the content from each project

Methods:

FavoritosController(Context context): Constructor that receives a context
Context context: Context view that the class uses.
FavoritosController(): Empty constructor.
void adicionar(ProjetoModel projeto, String conteudo): Writes a project at the archive
void remover(ProjetoModel projeto, String conteudo): Removes a project from the archive
String projetosEmString(): Transforms the content from all the projects into a string
void popularProjetos(String conteudoArquivo): Takes the content from the archive and transforms it into projects
void popularListaComProjetos(): Fills the project list to be shown
static ArrayList<ProjetoModel> getProjetosFavoritados(): Getters and Setters
static void setProjetosFavoritados(ArrayList<ProjetoModel> projetosFavoritados): Getters and Setters
static ArrayList<String> getProjetosFavoritadosCompletoStr(): Getters and Setters
static void setProjetosFavoritadosCompletoStr(ArrayList<String> projetosFavoritadosCompletoStr): Getters and Setters

Folder: Src
Package: com.mds.app.controller
Class: HistoricoController.java

Variables:

static ArrayList<ProjetoModel> projetosHistorico: Lists all the projects in the historic. Max: 10 projects.

static ArrayList<String> projetosFavoritadosCompletoStr: Lists all the data in the project at the historic string.

Persistencia persistencia: Class instance that communicates with the archive.

final int separadoresPorProjeto: Constant that counts how many separators there are in a project.

final int numeroDeProjetosNoArquivo: Counts how many projects there are in an archive.

int numeroDeSeparadores: Counts how many separators there are in an archive.

PartidoModel partido: Stores the data about political party.

ParlamentarModel parlamentar: Stores the data from a parlamentar.

ProjetoModel projeto: Stores the data from a project.

String stringProjeto: Temporary variable that stores the content of each project.

Methods:

HistoricoController(Context context): Constructor that receives a context.

Context context: View context that the class runs

HistoricoController(): Empty constructor

void adicionar(ProjetoModel projeto, String conteudo): Writes a project from the archive

void remover(ProjetoModel projeto, String conteudo): Removes a project from the archive

String projetosEmString(): Transforms all the content from projects into string

void popularProjetos(String conteudoArquivo): Takes the content from the archive and transforms it into projects

ArrayList<String> splitParts: List that stores the projects in string type

void popularListaComProjetos(): Fills the list of projects to be shown

static ArrayList<ProjetoModel> getProjetosHistorico(): Getters and Setters

static void setProjetosHistorico(ArrayList<ProjetoModel> projetosHistorico): Getters and Setters

static ArrayList<String> getProjetosHistoricoCompletoStr(): Getters and Setters

static void setProjetosHistoricoCompletoStr(ArrayList<String> projetosHistoricoCompletoStr): Getters and Setters

static int getNumeroDeProjetosNoHistorico(): Getters and Setters

static int getMaxProjetos(): Getters and Setters

static ProjetoModel getProjetoMaisVelho() throws NullPointerException: Getters and Setters

static String getStringProjetoMaisVelho() throws IndexOutOfBoundsException: Getters and Setters

Folder: Src
Package: com.mds.app.controller
Class: ListaControllerjava

Variables:

static final String SEPARADOR: Constant that represents a separator
static List<ProjetoModel> listaProjetos: List of projects to be shown at the screen
static ProjetoModel projetoAtual: Project that was clicked by user

Methods:

// Constructor from ListaController
- ListaController(List<ProjetoModel> result);

List<ProjetoModel> result: Receives a list to be shown by the search at the constructor

// Empty constructor
- ListaController();

// Transforms the list of projects into a string
- ArrayList<String> transformarLista()

ArrayList<String> stringProjetos: List of string to store content from projects

// Transform the list of projects into string with all the content
- String getStringCompletaParaPerfil()

String stringProjeto: stores the complete content from project of each iteration

// Transforms the list of projects into string with all the content and separators
String getStringCompletaParaArquivo()

String stringProjeto: Stores the complete content from the project

// Getters and Setters
- static List<ProjetoModel> getListaProjetos();
- static void setListaProjetos(List<ProjetoModel> novaListaProjetos);
- static ProjetoModel getProjetoAtual();
- static void setProjetoAtual(ProjetoModel projetoAtual);
- static String getSeparador();

Folder: Src
Package: com.mds.app.controller
Class: ProcuraParlamentarController.java

// Update the result values from the parlamentar search
- static void atualizarDadosPesquisaParlamentar(String nomeAutor);

String nomeAutor: Stores the parlamentar name for the seach

Folder: Src
Package: com.mds.app.controller
Class: ProcuraPartidoController.java

// Updates the result values from political party search
- static void atualizarDadosPesquisaPartdo(String uf, String siglaPartido);

String uf: Abreviation from the state name for the search

String siglaPartido: Abreviation from the parlamentar political party for the search

Folder: Src
Package: com.mds.app.controller
Class: ProcuraProjetoController.java

```
// Updates the values from the result of the project search
static void atualizarDadosPesquisaProjeto(String ano, String sigla, String numero, String dataIni);
```

String ano: Project year for the search. It must have 4 digits and be > 0

String sigla: Abreviation from the type of project for the search

String numero: Project number for the search. It must have 4 digits and be >0

String dataIni: Initial date of the project for the search. The format must be DD/MM/YYYY

Folder: Src
Package: com.mds.app.controller
Class: ProposicaoController.java

Variables:

StringBuffer buffer: Stores the states of search inside the xml
ArrayList<ProjetoModel> listaProjetos: List that stores the result projets from the search
ProjetoModel projeto : Actual project inside xml
ParlamentarModel parlamentar: Actual parlamentar inside xml
PartidoModel partido: Actual political party inside xml

Methods:

// Constructor that instantiate buffer
- ProposicaoController();

// Part of the parse API from XML
- void startElement(String namespaceURI, String localName, String qName, Attributes atts);

// Part from the parse API from XML
- void endElement(String uri, String localName, String qName) throws SAXException;

// Part from the parse API from XML
void characters(char[] ch, int start, int length)

// Getters and Setters
- ArrayList<ProjetoModel> getListaProjetos()
- StringBuffer getBuffer()
- void setBuffer(StringBuffer buffer)
- ProjetoModel getProjeto()
- void setProjeto(ProjetoModel projeto)
- ParlamentarModel getParlamentar()
- void setParlamentar(ParlamentarModel parlamentar)
- PartidoModel getPartido()
- void setPartido(PartidoModel partido)
- void setListaProjetos(ArrayList<ProjetoModel> listaProjetos)

2.2 com.mds.app.exception

Class ValidaEntrada

Methods:

// Date field validation
public static boolean validaData(String data)
String data: contains the date

// Autor field validation
public static boolean validaAutor(String autor)
String autor: contains the autor complete name

// Number field validation
public static boolean validaNumero(String numero)
String numero: contains the project number

// Year field validation
public static boolean validaAno(String ano)
String ano: contains start year of the Project

// Sigla field validation
public static boolean validaSigla(String sigla)
String sigla: contains Acronym for Project type

// UF field validation
public static boolean validaUf(String uf)
String uf: contains Acronym of the State

// Performs validation format all fields
public static boolean[] validandoEntradas(String ano, String sigla, String numero, String dataIni,String autor, String partido, String uf)
Same variables of the previous methods

// Gives preference to the result of parties, providing a server error
public static String garanteResultadoPartido(String uf, String partido)
Same variables of the previous methods

// Gives preference to the result of the author, providing a server error
public static String garanteResultadoAutor(String autor, String partido)
Same variables of the previous methods

// Creates the error message according to the form
public static String identificarErros(String ano, String sigla, String numero, String dataIni, String autor, String partido, String uf)
Same variables of the previous methods

2.3. com.mds.app.model

Folder: Model
Package: com.mds.app.model
Class: ParlamentarModel.java

Package:

com.mds.app.model: Package containing all the model elements from the application

Imports:

There is no imports in this class.

Class:

ParlamentarModel: It's a public model class for Parlamentar.

Variables:

nome: It's a private String type variable to store the name of the parliamentarian. It's accessed by the getter and setter methods.

partido: It's a private PatidoModel type variable to store the political party of the parliamentarian. It's accessed by the getter and setter methods.

Methods:

ParlamentarModel(): Default constructor method of the class.

ParlamentarModel(String nome, PartidoModel partido): This is the constructor method of the class that receives as parameters the a String variable and a PartidoModel variable.

getNome(): the public access method to verify the variable “nome” of the String type. It returns the “nome” variable value of the parliamentarian.

setNome(String nome): the public access method to modify the variable “nome” of the String type. It receives a String value as a parameter and stores it in the String variable “nome”. It's a void method, so it has no return.

getPartido(): the public access method to verify the variable “partido” of the PartidoModel type. It returns the “partido” variable value of the parliamentarian.

setPartido(PartidoModel partido): the public access method to modify the variable “partido” of the PartidoModel type. It receives a PartidoModel value as a parameter and stores it in the PartidoModel variable “partido”. It's a void method, so it has no return.

Folder: Model
Package: com.mds.app.model
Class: PartidoModel.java

Package:

com.mds.app.model: Package containing all the model elements from the application

Imports:

There is no imports in this class.

Class:

PartidoModel: It's a public model class for Partido.

Variables:

siglaPartido: It's a private String type variable to store the acronym of the political party. It's accessed by the getter and setter methods.

uf: It's a private PatidoModel type variable to store the “uf” (acronym for Federation Unity) of the political party. It's accessed by the getter and setter methods.

Methods:

PartidoModel(): Default constructor method of the class.

PartidoModel(String siglaPartido, String uf): This is the constructor method of the class that receives as parameters the a String variable for “siglaPartido” and a String variable for “uf”.

getSiglaPartido(): the public access method to verify the variable “siglaPartido” of the String type. It returns the “siglaPartido” variable value of the political party.

setSiglaPartido(String siglaPartido): the public access method to modify the variable “siglaPartido” of the String type. It receives a String value as a parameter and stores it in the String variable “siglaPartido”. It's a void method, so it has no return.

getUf(): the public access method to verify the variable “uf” of the String type. It returns the “uf” variable value of the political party.

setUf(String uf): the public access method to modify the variable “uf” of the String type. It receives a String value as a parameter and stores it in the String variable “uf”. It's a void method, so it has no return.

Folder: Model
Package: com.mds.app.model
Class: ProcuraParlamentarModel.java

Package:

com.mds.app.model: Package containing all the model elements from the application

Imports:

There is no imports in this class.

Class:

ProcuraParlamentarModel: It's a public abstract model class for ProcuraParlamentar, which means you can not instantiate an object.

Variables:

nome: It's a private and static String type variable to store the name of the parliamentarian you want to search. Since this variable is static, you do not need to instantiate an object to access it.

Methods:

getNome(): the public and static access method to verify the variable “nome” of the String type. It returns the “nome” variable value of the parliamentarian you want to search.

setNome(String nome): the public and static access method to modify the variable “nome” of the String type. It receives a String value as a parameter and stores it in the String variable “nome”. It's a void method, so it has no return.

Folder: Model
Package: com.mds.app.model
Class: ProcuraPartidoModel.java

Package:

com.mds.app.model: Package containing all the model elements from the application

Imports:

There is no imports in this class.

Class:

ProcuraPartidoModel: It's a public abstract model class for ProcuraParlamentar, which means you can not instantiate an object.

Variables:

uf: It's a private and static String type variable to store the “uf” (acronym for Federal Unity) of the political party you want to search. Since this variable is static, you do not need to instantiate an object to access it.

sigla: It's a private and static String type variable to store the acronym of the political party you want to search. Since this variable is static, you do not need to instantiate an object to access it.

Methods:

getUf(): the public and static access method to verify the variable “uf” of the String type. It returns the “uf” variable value of the political party you want to search.

setUf(String uf): the public and static access method to modify the variable “uf” of the String type. It receives a String value as a parameter and stores it in the String variable “uf”. It's a void method, so it has no return.

getSigla(): the public and static access method to verify the variable “sigla” of the String type. It returns the “sigla” variable value of the political party you want to search.

setSigla(String sigla): the public and static access method to modify the variable “sigla” of the String type. It receives a String value as a parameter and stores it in the String variable “sigla”. It's a void method, so it has no return.

Folder: Model
Package: com.mds.app.model
Class: ProcuraProjetoModel.java

Package:

com.mds.app.model: Package containing all the model elements from the application

Imports:

There is no imports in this class.

Class:

ProcuraProjetoModel: It's a public abstract model class for ProcuraProjeto, which means you can not instantiate an object.

Variables:

ano: It's a private and static String type variable to store the year of the project you want to search. Since this variable is static, you do not need to instantiate an object to access it.

id: It's a private and static String type variable to store the ID of the project you want to search. Since this variable is static, you do not need to instantiate an object to access it.

sigla: It's a private and static String type variable to store the acronym of the of the political party who emited the project you want to search. Since this variable is static, you do not need to instantiate an object to access it.

dataInicio: It's a private and static String type variable to store the emission date of the project you want to search. Since this variable is static, you do not need to instantiate an object to access it.

Methods:

getAno(): the public and static access method to verify the variable “ano” of the String type. It returns the “ano” variable value of the project you want to search.

setAno(String ano): the public and static access method to modify the variable “ano” of the String type. It receives a String value as a parameter and stores it in the String variable “ano”. It's a void method, so it has no return.

getId(): the public and static access method to verify the variable “id” of the String type. It returns the “id” variable value of the project you want to search.

setId(String id): the public and static access method to modify the variable “id” of the String type. It receives a String value as a parameter and stores it in the String variable “id”. It's a void method, so it has no return.

getSigla(): the public and static access method to verify the variable “sigla” of the String type. It returns the “sigla” variable value of political party who emited the project you want to search.

setSigla(String sigla): the public and static access method to modify the variable “sigla” of the String type. It receives a String value as a parameter and stores it in the String variable “sigla”. It's a void method, so it has no return.

getDataInicio(): the public and static access method to verify the variable “dataInicio” of the String type. It returns the “dataInicio” variable value of the emission date of the project you want to search.

setDataInicio(String dataInicio): the public and static access method to modify the variable “dataInicio” of the String type. It receives a String value as a parameter and stores it in the String variable “dataInicio”. It's a void method, so it has no return.

Folder: Model
Package: com.mds.app.model
Class: ProjetoModel.java

Package:
com.mds.app.model: Package containing all the model elements from the application

Imports:
There is no imports in this class.

Class:
ProjetoModel: It's a public model class for “Projeto”.

Variables:
id: It's a private String type variable to store the ID of the project. It's accessed by the getter and setter methods.
ano: It's a private String type variable to store the year of the project . It's accessed by the getter and setter methods.
numero: It's a private String type variable to store the number of the project. It's accessed by the getter and setter methods.
// **nome:** It's a private String type variable to store the name of the parliamentarian who emited the project. It's accessed by the getter and setter methods.
sigla: It's a private String type variable to store the acronym of the political party. It's accessed by the getter and setter methods.
data: It's a private String type variable to store the emission date of the project. It's accessed by the getter and setter methods.
explicacao: It's a private String type variable to store the the explanation of the project. It's accessed by the getter and setter methods.
status: It's a private String type variable to store the actual status of the project. It's accessed by the getter and setter methods.
parlamentar: It's a private ParlamentarModel type variable to store the parliamentarian responsable for the project. It's accessed by the getter and setter methods.
cont: It's a private interger type variable used to set a position in the XML file. It's accessed by the getter and setter methods. It's initial value is 0 (zero).
contId: It's a private interger type variable used to set a position in the desired ID in the XML file. It's accessed by the getter and setter methods. It's initial value is 0 (zero).

Methods:
ProjetoModel(): Default constructor method of the class.
PartidoModel(String ano, String nome, String sigla, String data, String numero, String explicacao, ParlamentarModel parlamentar): This is the constructor method of the class that receives as parameters a String variable for “ano”, a String variable for “nome”, a String variable for “sigla”, a String variable for “data”, a String variable for “numero”, a String variable for “explicacao” and a ParlamentarModel variable for “parlamentar”. It also sets the value 1 (one) the “cont” variable.
getAno(): the public access method to verify the variable “ano” of the String type. It returns the “ano” variable value of the emission date of the project.
setAno(String Ano): the public access method to modify the variable “ano” of the String type. It receives a String value as a parameter and stores it in the String variable “ano”. It's a void method, so it has no return.
getNumero(): the public access method to verify the variable “numero” of the String type. It returns the “numero” variable value of number of the project.

setNumero(String numero): the public access method to modify the variable “numero” of the String type. It receives a String value as a parameter and stores it in the String variable “numero”. It's a void method, so it has no return.

getNome(): the public access method to verify the variable “nome” of the String type. It returns the “nome” variable value of the parliamentarian who emited the project.

setNome(String nome): the public access method to modify the variable “nome” of the String type. If the counting variable “cont” iquals 1 (one) it receives a String value as a parameter and stores it in the String variable “nome”, else if “cont” is different than 1 (one), it does nothing. In the end of the method, the counting variable “cont” is auto-increased. It's a void method, so it has no return.

setId(String id): the public access method to modify the variable “id” of the String type. If the counting variable “contId” iquals 0 (zero) it receives a String value as a parameter and stores it in the String variable “id” and the counting variable “contId” is auto-increased, else if “contId” is different than 0 (zero), it does nothing. It's a void method, so it has no return.

getSigla(): the public access method to verify the variable “sigla” of the String type. It returns the “sigla” variable value of the political party.

getId(): the public access method to verify the variable “id” of the String type. It returns the “id” variable value of the project.

setSigla(String sigla): the public access method to modify the variable “sigla” of the String type. It receives a String value as a parameter and stores it in the String variable “sigla”. It's a void method, so it has no return.

getData(): the public access method to verify the variable “data” of the String type. It returns the “data” variable value of the emission date of the project.

setData(String data): the public access method to modify the variable “data” of the String type. It receives a String value as a parameter and stores it in the String variable “data”. It's a void method, so it has no return.

getExplicacao(): the public access method to verify the variable “explicacao” of the String type. It returns the “explicacao” variable value of explanation of the project.

setExplicacao(String explicacao): the public access method to modify the variable “explicacao” of the String type. It receives a String value as a parameter and stores it in the String variable “explicacao”. It's a void method, so it has no return.

getParlamentar(): the public access method to verify the variable “parlamentar” of the ParlamentarModel type. It returns the “parlamentar” variable value of the parliamentarian who emited the project.

setParlamentar(ParlamentarModel parlamentar): the public access method to modify the variable “parlamentar” of the ParlamentarModel type. It receives a String value as a parameter and stores it in the ParlamentarModel variable “parlamentar”. It's a void method, so it has no return.

getCont(): the public access method to verify the variable “cont” of the interger type. It returns the “cont” counting variable value.

setCont(int cont): the public access method to modify the variable “cont” of the interger type. It receives a interger value as a parameter and stores it in the interger variable “cont”. It's a void method, so it has no return.

getContId(): the public access method to verify the variable “contId” of the interger type. It returns the “contId” counting ID variable value.

setContId(String contId): the public access method to modify the variable “contId” of the interger type. It receives a interger value as a parameter and stores it in the interger variable “contId”. It's a void method, so it has no return.

getStatus(): the public access method to verify the variable “status” of the String type. It returns the “status” variable value of the status of the project.

setStatus(String status): the public access method to modify the variable “status” of the String type. It receives a String value as a parameter and stores it in the String variable “status”. It's a void method, so it has no return.

toString(): a public method with the @Override anotation that represent one ProjetoModel object as a String. It returns a String with all the information of the constructed object.

2.4. com.mds.app.persistencia

2.5. com.mds.app.services

Folder: Services
Package: com.mds.app.services
Class: Endereco.java

Package:

com.mds.app.services: Package containing all the conection services elements of the aplication

Imports:

There is no imports in this class.

Class:

Endereco: It's a public abstract service class for the governamental website address. It's basically a class that contains many final (constant) variables and can not be instantiated beacause it's declared as abstract. It constructs a URL that can be accessed by other classes, beacause all the variables and methods are declared as static.

Variables:

BASE_URL: It's a private final static String type variable. This variable receives the base URL of the governamental site “www.camara.gov.br” .

IGUAL: It's a private final static String type variable. This variable receives the character “=”, used to construct the URL.

E: It's a private final static String type variable. This variable receives the character “&”, used to construct the URL.

SIGLA: It's a private final static String type variable. This variable receives the string from the variable “sigla”, which is the acronym of the political party used to construct the URL.

NUMERO: It's a private final static String type variable. This variable receives the string from the variable “numero”, which is the number of the project used to construct the URL.

ANO: It's a private final static String type variable. This variable receives the string from the variable “ano”, which is the the actual year used to construct the URL.

DATA_INICIO: It's a private final static String type variable. This variable receives the string from the variable “dataInicio”, which is the date the project was emited used to construct the URL.

DATA_FINAL: It's a private final static String type variable. This variable receives the string from the variable “dataFinal”, which is the final date of the project, used to construct the URL.

AUTOR: It's a private final static String type variable. This variable receives the string from the variable “autor”, which is the author of the project, used to construct the URL.

NOME_AUTOR: It's a private final static String type variable. This variable receives the string from the variable “nomeAutor”, which is the name of the author of the project used to construct the URL.

SIGLA_PARTIDO: It's a private final static String type variable. This variable receives the string from the variable “siglaPartido”, which is the political party acronym used to construct the URL.

SIGLA_UF: It's a private final static String type variable. This variable receives the string from the variable “siglaUF”, which is the acronym of the Federal Unity (“Unidade Federativa”) used to construct the URL.

GENERO_AUTOR: It's a private final static String type variable. This variable receives the string from the variable “generoAutor”, which is the gender of the author of the project used to construct the URL.

CODIGO_ESTADO: It's a private final static String type variable. This variable receives the string from the variable “codigoEstado”, which is the state code used to construct the URL.

CODIGO_ORGAO_ESTADO: It's a private final static String type variable. This variable receives the string from the variable “codigoOrgaoEstado”, which is the state organ code used to construct the URL.

TRAMITACAO: It's a private final static String type variable. This variable receives the string from the variable “tramitacao”, which is the processing state of the project used to construct the URL.

sigla: It's a public static String type variable. This variable receives an acronym data string and it's used to construct de URL.

numero: It's a public static String type variable. This variable receives a number data string and it's used to construct de URL.

ano: It's a public static String type variable. This variable receives a year string data and it's used to construct de URL.

dataInicio: It's a public static String type variable. This variable receives an initial date data string and it's used to construct de URL.

dataFinal: It's a public static String type variable. This variable receives a final date data string and it's used to construct de URL.

autor: It's a public static String type variable. This variable receives an author data string and it's used to construct de URL.

nomeAutor: It's a public static String type variable. This variable receives an author name data string and it's used to construct de URL.

siglaPartido: It's a public static String type variable. This variable receives a political party acronym data string and it's used to construct de URL.

siglaUF: It's a public static String type variable. This variable receives a Federal Unity (“Unidade Federativa”) acronym data string and it's used to construct de URL.

generoAutor: It's a public static String type variable. This variable receives an author gender data string and it's used to construct de URL.

codigoEstado: It's a public static String type variable. This variable receives a state code data string and it's used to construct de URL.

codigoOrgaoEstado: It's a public static String type variable. This variable receives a state organ code data string and it's used to construct de URL.

ID_TIPO_AUTOR: It's a private final static String type variable. This variable receives the string from the variable “idTipoAutor”, which is the id type of the author used to construct the URL.

Methods:

contruirEndereco(): this method receives all the information of the public static String variables described above, which is needed to construct the URL. The return is the URL to a toString() method, that will create the final URL.

Folder: Src
Package: com.mds.app.services
Class: RecebeHTTP.java

Package:

com.mds.app.services: Package containing all the elements from Services.

Imports:

java.io.BufferedReader: To read texts form input stream. Usually used to read files.
java.io.IOException: To throw input/output exceptions
java.io.InputStreamReader: A bridge from byte streams to character streams. Usually used to read files.
java.net.URI: A uniform resource identifier that identifies an abstract or physical resource.
java.net.URISyntaxException: A java.net.URISyntaxException will be thrown if some information could not be parsed while creating a URI.
org.apache.http.HttpResponse: After receiving a request message, a server responds with an HTTP responde message.
org.apache.http.client.ClientProtocolException: An exception in case of error establishing the client protocol.
org.apache.http.client.HttpClient: Interface for an HTTP client. HTTP clients encapsulate a smorgasbord of objects required to execute HTTP requests while handling cookies, authentication, connection management, and other features. Thread safety of HTTP clients depends on the implementation and configuration of the specific client.
org.apache.http.client.methods.HttpGet: The GET method means retrieve whatever information (in the form of an entity) is identified by the Request-URI.
org.apache.http.impl.client.DefaultHttpClient: Default implementation of an HTTP client.

Class:

RecebeHTTP: Class responsible to receive and HTTP address.

Variables:

BufferedReader inputStream: It will receive the concent from an entitity.
String dado: it will receive the http.
HttpClient cliente: An instance from DefaultHttpClient object.
HttpGet requisicao: It starts the requisition and sets the URI.
HttpResponse resposta: It receives the executation from DefaultHttpClient.
URI website: The URI
StringBuffer dadoStringBuffer: Receives what was anexed.
String anexar: Receives something that must be annexed to the dado.
String url: an url.

Methods:

public recebeHTTP(): Only an empty constructor.
public String recebe(String url): Receives an url and returns the dado.
new URI(url): returns a new URI by an URL.
printStackTrace(): Returns the error/exception.
httpGet(): The GET method means retrieve whatever information (in the form of an entity) is identified by the Request-URI.
requisicao.setURI(website): it sets an URI based in an URL.
cliente.execute(requisicao): defines an executation.
new BufferedReader(new InputStreamReader(resposta.getEntity().getContent())):

Reads the content from an entity and stores it in a variable.

dadoStringBuffer.append(anexar): Append the param content to the dadoStringBuffer.

dadoStringBuffer.toString(): Returns a string with the content.

3. Jogo do Morro

Audio.java

Class Variables:

Clip clip: It loads and play the audio
String fileName: Variable that keeps the audio's file path that will be loaded

Methods:

// It loads and play the audio
- void tocarMusica();

File file: It loads the file

// Getters and Setters

- String getFileName();
- void setFileName(String fileName);

BufferedImageLoader.java

Class Variables:

BufferedImage image: It keeps the loaded image

Methods:

// Load a image
- BufferedImage loadImage(String path) throws IOException;

String path: where is the image to be loaded

Cenario.java

Class Variables:

int altura: height in pixels of the scene, always positive
int largura: width in pixels of the scene, always positive

Methods:

// Constructor
- Cenario(int largura, int altura);

// Getters and Setters
- int getAltura();
- void setAltura(int altura);
- int getLargura();
- void setLargura(int largura) ;

Colisao.java

Methods:

// Check colision between player and enemies
- boolean colisao(Player a, Inimigo[] b);

// Check colision between player and itens
- boolean colisaoItem(Player a, Item[] b);

// Check colision between player's shoot and enemies
- int colisaoTiro(Player a, Inimigo[] b);

//It is never used!
boolean colisaoSeta(Player a, Rectangle b);

// // Check colision between player and enemies's shoot
boolean colisaoTiroPlayer(Player a, Inimigo[] b);

Fase.java

Class variables:

int altura: height in pixels of the level, always positive
int largura: width in pixels of the level, always positive
boolean completo: It says if the player have passed the level
int qtd_plataforma: Number of platforms in a level
Inimigo inimigo[]: A vector with all the enemies in the level
Player player: A reference of the player
Cenario cenario: It has the scene atributes

Methods:

// Empty constructor, it sets everything 0
- Fase();

// Constructor with all the parameters being set
- Fase(int altura, int largura, boolean completo, int qtd_plataforma, Inimigo[] inimigo, Player player, Cenario cenario);

// Getters and Setters
- int getAltura();
- void setAltura(int altura);
- int getLargura();
- void setLargura(int largura);
- boolean getCompleto();
- void setCompleto(boolean completo);
- int getQtd_plataforma();
- void setQtd_plataforma(int qtd_plataforma);
- Inimigo[] getInimigos();
- void setInimigos(Inimigo[] inimigo);
- Player getPlayer();

- void setPlayer(Player player);
- Cenario getCenario();
- void setCenario(Cenario cenario);

Item.java

Class variables

Game game: A reference to the Game
BufferedImage item: Loaded image of the item

Methods:

Constructor with all the parameters being set
- Item(int x, int y, int altura, int largura, Game game);

// Returns the dimensions of the image
- Rectangle getBounds();

// Render the item in the screen
- void render(Graphics g);

Graphics g: render context of the screen

Keyboard.java

Class variables:

Game game: A reference to the Game

Methods:

// Constructor
- Keyboard(Game game)

// Returns the value of the pressed key
- void keyPressed(KeyEvent e)

// Returns the value of the released key
void keyReleased(KeyEvent e)

Mouse.java

Class variables:

int mouseX: mouse X position
int mouseY: mouse Y position

Methods:

// Returns the value of the mouse in the X axis
- static int getX();

// Returns the value of the mouse in the Y axis
- static int getY();

// Altera os valores de X e Y do mouse quando um clique é efetuado
void mousePressed(MouseEvent e);

// Setters
- void setMouseX(int mouseX)
- void setMouseY(int mouseY)

Pessoa.java

Class variables:

float x: position related to the X axis, sometimes it is not positive
float y: position related to the Y axis, sometimes it is not positive
int altura: height in pixels of the image that represents the person
int largura: width in pixels of the image that represents the person
boolean atirar: state where a person can shoot
boolean esquerda: state where the person is walking to the left
boolean direita: state where the person is walking to the right

Methods:

// Constructor with all the parameters being set
- Pessoa(int x, int y, int altura, int largura, boolean atirar, boolean esquerda, boolean direita);

// The person shoots
- void atirar();

// Getters and Setters
- boolean isEsquerda();
- void setEsquerda(boolean esquerda);
- boolean isDireita();
- void setDireita(boolean direita);
- float getX();
- void setX(float x);
- float getY();
- void setY(float y);
- int getAltura();
- void setAltura(int altura);
- int getLargura();
- void setLargura(int largura);
- boolean isAtirar();
- void setAtirar(boolean atirar);

plataforma.java

Variaveis da classe:

int x: X position of the platform, always positive
int y: Y position of the platform, always positive
int largura: width of the platform,always positive

Métodos:

// Constructor with all the parameters being set
- plataforma(int x, int y, int largura);

// Getters and Setters
- int getX();
- void setX(int x);
- int getY();
- void setY(int y);
- int getLargura();
- void setLargura(int largura);

SpriteSheet.java

Class variables:

BufferedImage image: image to be drawn

Methods:

// Constructor that sets the image
- SpriteSheet(BufferedImage image);

// Returns part of the sprite's image to be rendered
BufferedImage grabImage(int col, int row, int width, int height);

int col: column in the spritesheet
int row: row in the spritesheet
int width: width of the image to be cut
int height: height of the image to be cut

Inimigo.java

Class variables:

Game game: reference to the game
BufferedImage inimigo: Enemy's image, store the loaded image from the resourses
int limite: limit to the left that the enemy can walk
int limite2: limit to the right that the enemy can walk
boolean morto; if the enemy is dead this variable is true
int tipo: the type of enemy (type 1 walks from a side to another, type 2 just stay in the same place)
int direcao : which direction the enemy is walking (only applies if the type is 1)
float xtiro: X position of the shoot
float ytiro: Y position of the shoot
Audio audio: enemy's death sound

Methods:

// Constructor with all the parameters being set
- Inimigo(int x, int y, int altura, int largura, Game game, int tipo);

// Sets the enemy's death sound
- void somDeMorte();

// Render the enemy and CALCULATES THE SHOOT POSITION
- void render(Graphics g);

// Render the shoot
- void atirar(float x, float y);

// Return the position and size of the shoot
- Rectangle getTiroBounds();

// Calculates the enemy's positions (IA)
- void tick();

int cont3: counter to the enemy's walking algorithm

// Returns enemy's position and size
- Rectangle getBounds();

// Getters and Setters
- void setImage();
- float getX();
- void setX(float x);
- float getY();
- void setY(float y);
- int getLimite();
- void setLimite(int limite);
- int getLimite2();
- void setLimite2(int limite2);
- int getTipo();
- void setTipo(int tipo);
- boolean isMorto();
- void setMorto(boolean morto);
- float getXtiro();
- void setXtiro(float xtiro);
- float getYtiro();
- void setYtiro(float ytiro);
- int getDirecao();
- void setDirecao(int direcao);

Player.java

Class variables:

plataforma[] plat: vector with all the platforms of the level
boolean plataforma: if the player is over the platform it gets true
boolean chao: if the player is over the ground it gets true
boolean saltar: if the player is in a jump it gets true
int posicao: position of the image in the spritesheet
Game game: reference to the game
float xtiro: player's shoots position in the X axis
float ytiro: player's shoots position in the Y axis
boolean permiteTiro: it is true when the player can shoot (once by level)
int i: it is never used
BufferedImage player: player's loaded image
int direcao: direction that the player is moving (0 > stoped, 1 > left, 2 >right)
boolean morreu: it gets true when the player dies
Audio audio: store the player's death sound

Methods:

// Constructor with all the parameters being set
- Player(int x, int y, int altura, int largura, Game game);

// Update the player's position, and verify if he is over a platform or over the ground, jump when it is possible
- void tick();

// Render the player, verifies if shoot is available e calculates the shoot's position
- void render(Graphics g);

// Returns the player's position and size
- Rectangle getBounds();

// Returns the bullet's position and size
- Rectangle getTiroBounds();

//It plays the shoot's sound
- void somDeTiro();

// Render the bullet
- void atirar(float x, float y);

// Getters and Setters
- boolean getSaltar();
- void setSaltar(boolean saltar);
- plataforma[] getPlat();
- boolean isplataforma();
- void setplataforma(boolean plataforma);
- void setPlat(plataforma plat[]);
- int getAltura();
- void setAltura(int altura);

- int getLargura();
- void setLargura(int largura);
- Game getGame();
- void setGame(Game game);
- float getXtiro();
- void setXtiro(float xtiro);
- float getYtiro();
- void setYtiro(float ytiro);
- boolean isPermiteTiro();
- void setPermiteTiro(boolean permiteTiro);
- int getDirecao();
- void setDirecao(int direcao);
- boolean isMorreu();
- void setMorreu(boolean morreu);

Game.java

Class variables:

static final int WIDTH: constant width of the screen
static final int HEIGHT: constant height of the screen
static final int SCALE: scale to screen resolution
final String TITLE: constant fot the title of the game
float gravidade: value for the gravity
float velocidade: speed of the player in the formula of gravity (the axis y)
boolean running: game state, true if they run
Thread thread: thread for the game
BufferedImage image: scene's image
BufferedImage imageitem: item's image
BufferedImage spriteSheet: player's image
BufferedImage inimigoImage: type 1 enemy's image
BufferedImage inimigoImage2: type 2 enemy's image
BufferedImage menu: menu image
BufferedImage tiro: shot image
BufferedImage intro1: introduction image 1
BufferedImage intro2: introduction image 2
BufferedImage intro3: introduction image 3
BufferedImage intro[] = images of introduction array
BufferedImage setaEsquerda: Left signalization image
BufferedImage setaDireita: Right signalization image
BufferedImage raquel: lady’s image
BufferedImage fim: ending game image
int introducao: couting for the introduction animation
Player p: player
plataforma chao: platform that occupies the position of the ground
plataforma plat: platform of the game
plataforma plat2: platform of the game
plataforma plat3: platform of the game
plataforma plat4: platform of the game
plataforma plat5: platform of the game
plataforma plat6: platform of the game
plataforma plat7: platform of the game

plataforma plat8: platform of the game
plataforma plat9: platform of the game
plataforma plat10: platform of the game
plataforma[] plats: array of platforms of the game
Inimigo inimigo1: an enemy of the game
Inimigo inimigo2: an enemy of the game
Inimigo inimigo3: an enemy of the game
Inimigo inimigo4: an enemy of the game
Inimigo inimigo5: an enemy of the game
Inimigo inimigo6: an enemy of the game
Inimigo inimigo[]: enemies array
Item item1: game item
Item item2: game item
Item item3: game item
Item[] item: itens array
boolean renderizaitem[] : array that lets you render items
Audio audio: audio for the music
int faseAtual: counting for the actual stage, positive maximum 4
int faseAntiga: counting for the last stage , positive maximum 3
BufferedImageLoader loader: to load the images
int o: never used

Methods:

// load the sound of the game
- void somDeFundo();

//Initializes the game (Loads the images of the game, the enemies defines defines the platforms and instantiates the keyboard and mouse)
- void init();

// starts the thread of the game
- synchronized void start();

// stop the thread of the game
- synchronized void stop();

// Do the infinite loop, where the game runs
- void run();

long lastTime: last time since the update
final double amountOfTickets: Maximum number of frames
double ns: time between frames
double delta: accumulated time between the last frame and the current time
int updates: The counter of updates between one frame and another
int frames: Counter of frames
long timer: Current time
long now: Current time in loop of the game

// Updates scenery, checks collisions in the game, updates position and amount of enemies, upgrades position and number of platforms to suit each stage, and determines if the player went from stage.

void tick();

// Renders all graphics elements in the game
- void render();

// Calculate and apply the player gravity
- void aplicarGravidade(Player player);

// Checks collision between player and enemies
- boolean colisao(Player a, Inimigo[] b);

// Checks collision between player and itens
- boolean colisaoItem(Player a, Item[] b);

// Checks collision between the player shot and enemies
- int colisaoTiro(Player a, Inimigo[] b);

// Checks collision between the player and the enemies shots
boolean colisaoTiroPlayer(Player a, Inimigo[] b);

// Activates the corresponding actions when a certain key is pressed
- void keyPressed(KeyEvent e);

// Disables the corresponding actions when a certain key is released
- void keyReleased(KeyEvent e);

// Start of the game
- void main(String[] args);

Game game: all the game
JFrame frame: frame where the game appears

// Getters and Setters
- BufferedImage getSpriteSheet();
- BufferedImage getInimigoImage();
- BufferedImage getInimigoImage2();
- void setInimigoImage(BufferedImage inimigoImage);
- float getVelocidade();
- void setVelocidade(float velocidade);
- BufferedImage getImageitem();
- void setImageitem(BufferedImage imageitem);
- BufferedImage getTiroImage();