MANUAL TÉCNICO

SIMULADOR DE AEROPUERTO COMO PROYECTO DE IPC1: APLICACIÓN DE ESCRITORIO EN JAVA

HERRAMIENTAS UTILIZADAS

● LENGUAJE DE PROGRAMACIÓN JAVA - VERSIÓN 11

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems.

El lenguaje Java proporciona:

- El paradigma de la programación orientada a objetos.
- Ejecución de un mismo programa en múltiples sistemas operativos y plataformas.
- Es posible utilizarlo para múltiples propósitos, desde aplicaciones de escritorio hasta en servidores web.
- Tiene una curva de aprendizaje media pero también toma lo mejor de otros lenguajes orientados a objetos, como C++.

● IDE: APACHE NETBEANS 14

NetBeans es un entorno de desarrollo integrado, de código abierto y sigue la filosofía del software libre, y está hecho principalmente para el uso del lenguaje de programación Java. También existen muchísimos módulos extras para extender su funcionamiento. NetBeans es un producto libre y gratuito sin restricciones de uso y es un proyecto de código abierto de gran éxito con una gran base de usuarios y una comunidad en constante crecimiento.

● GITHUB

Es un sistema de control de versiones de código y gestión de proyectos, a su vez también funciona como una plataforma de estilo red social diseñada para desarrolladores para poder compartir código entre más personas y colaborar en el mismo.

- ## DIAGRAMS.NET - DRAW.IO

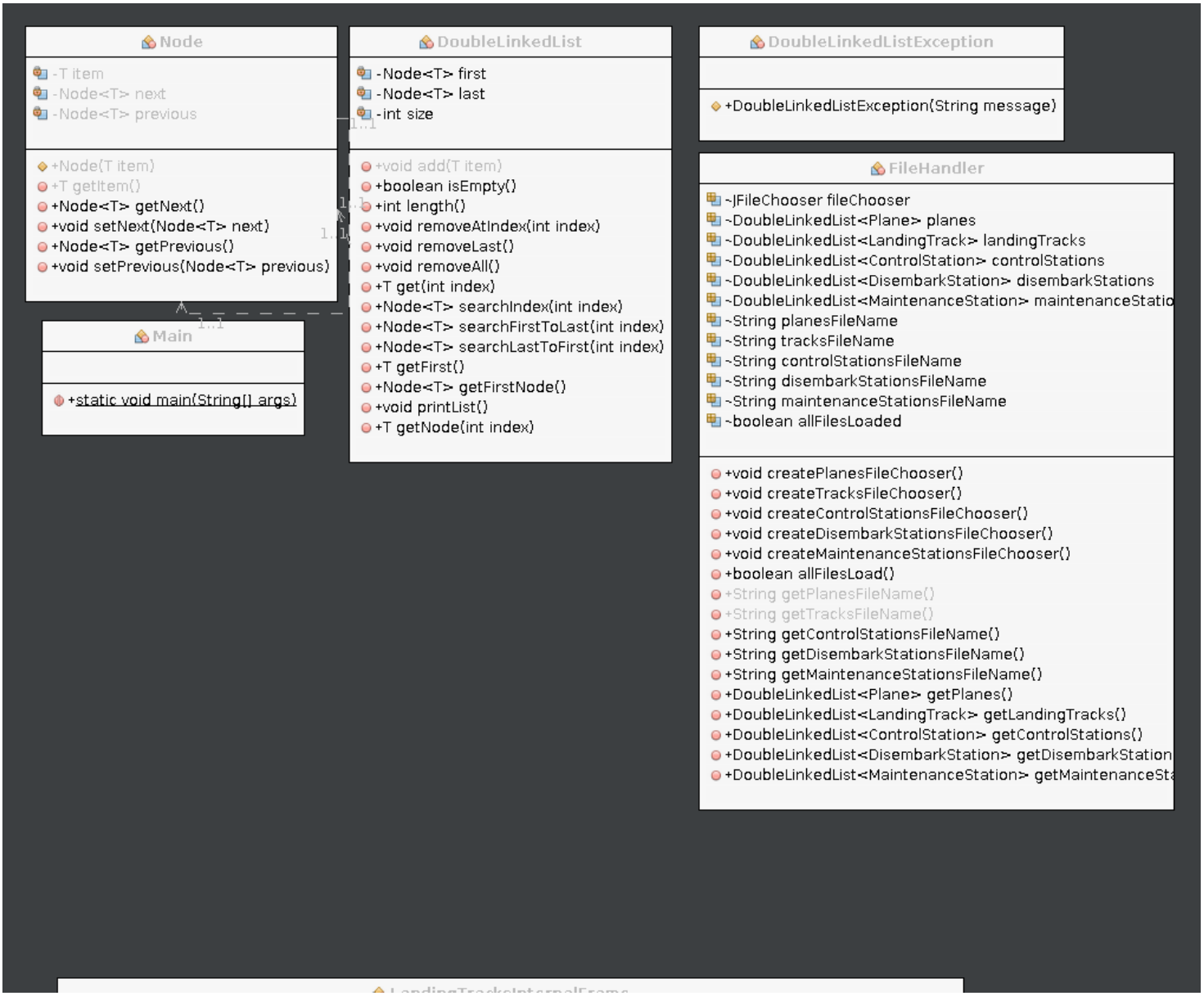
Draw.io es una herramienta de creación y edición de diagramas libre que permite la integración con diversas plataformas, principalmente Google Drive, el cual permite la compartición y colaboración de varias personas dentro del mismo proyecto. El software consiste en una aplicación web realizada mayoritariamente en JavaScript y licenciada con Apache v2, lo que la hace funcionar en una amplia gama de navegadores y permite la creación de diagramas, contando con modelos para diversos tipos como pueden ser diagramas UML, esquemas de red, flujogramas, etc. También permite crear colecciones de diagramas e imágenes personalizados para utilizar en los diagramas.

- ## FEDORA LINUX 36 - SISTEMA OPERATIVO

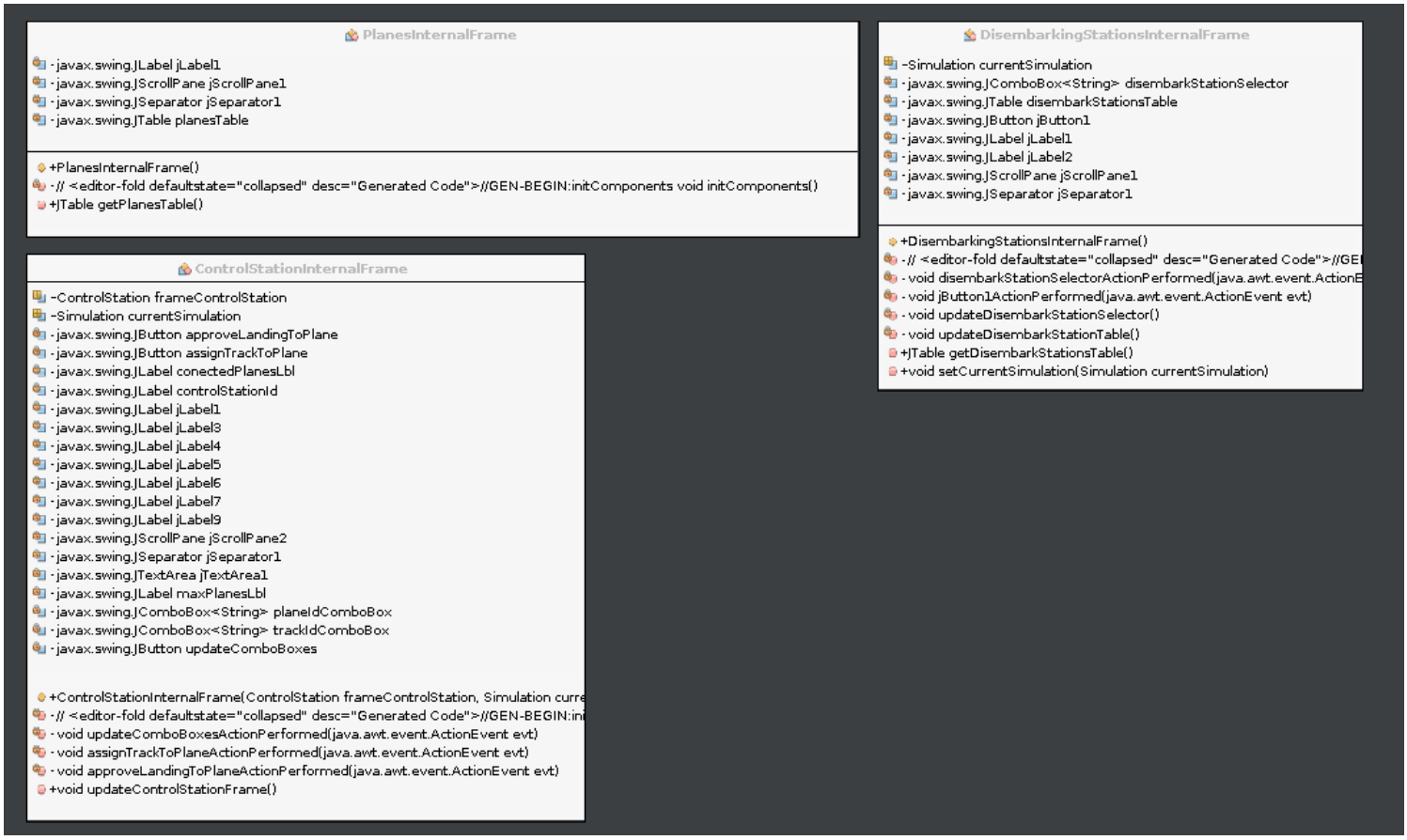
Fedora Linux es una distribución GNU/Linux para propósitos generales basada en RPM, y es mantenida por una comunidad internacional de ingenieros, diseñadores gráficos y usuarios. Se caracteriza por su estabilidad, innovación y contar con el apoyo y patrocinio de RedHat (subsidiaria de IBM). El proyecto no busca sólo incluir software libre y de código abierto, sino ser el líder en ese ámbito tecnológico.

DIAGRAMA DE CLASES

PAQUETE: `utils`



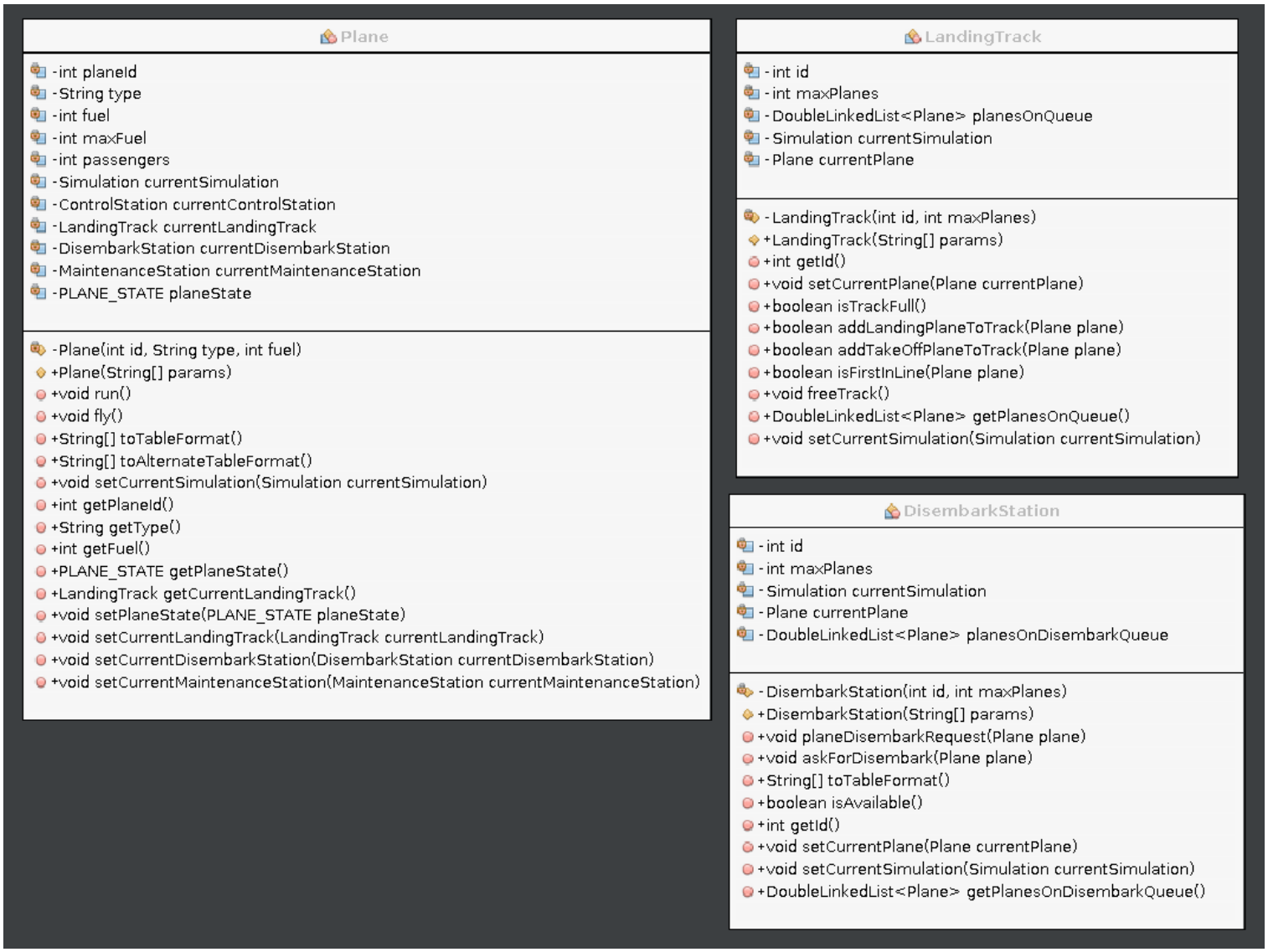
PAQUETE: frames




MaintenanceStationsInternalFrame
-Simulation currentSimulation
-javax.swing.JButton jButton1
-javax.swing.JLabel jLabel1
-javax.swing.JLabel jLabel2
-javax.swing.JScrollPane jScrollPane1
-javax.swing.JSeparator jSeparator1
-javax.swing.JComboBox<String> maintenanceStationSelector
-javax.swing.JTable maintenanceStationsTable
+MaintenanceStationsInternalFrame()
-// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents void
-void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
-void maintenanceStationSelectorActionPerformed(java.awt.event.ActionEvent evt)
+void updateMaintenanceStationSelector()
-void updateMaintenanceStationTable()
+JTable getMaintenanceStationsTable()
+void setCurrentSimulation(Simulation currentSimulation)
LoadFilesDialog
-FileHandler fileHandler
-int CONSUME_FUEL_TIME
-int LANDING_TIME
-int DISEMBARKING_TIME
-int MAINTENANCE_TIME
-int TAKE_OFF_TIME
-MainFrame mainFrame
-javax.swing.JButton continueBtn
-javax.swing.JButton controlStationsFileBtn
-javax.swing.JLabel controlStationsFileNameLbl
-javax.swing.JButton disembarkingStationsFileBtn
-javax.swing.JLabel disembarkingStationsFileNameLbl
-javax.swing.JLabel jLabel1
-javax.swing.JLabel jLabel10
-javax.swing.JLabel jLabel11
-javax.swing.JLabel jLabel13
-javax.swing.JLabel jLabel14
-javax.swing.JLabel jLabel16
-javax.swing.JLabel jLabel17
-javax.swing.JLabel jLabel19
-javax.swing.JLabel jLabel2
-javax.swing.JLabel jLabel20
-javax.swing.JLabel jLabel3
-javax.swing.JLabel jLabel4
-javax.swing.JLabel jLabel5
-javax.swing.JLabel jLabel6
-javax.swing.JLabel jLabel7
-javax.swing.JLabel jLabel9
-javax.swing.JSeparator jSeparator1
-javax.swing.JSeparator jSeparator2
-javax.swing.JSeparator jSeparator3
-javax.swing.JSeparator jSeparator4
-javax.swing.JSeparator jSeparator5
-javax.swing.JButton maintenanceStationsFileBtn
-javax.swing.JLabel maintenanceStationsFileNameLbl
-javax.swing.JButton planesFileBtn
-javax.swing.JLabel planesFileNameLbl
-javax.swing.JButton tracksFileBtn
-javax.swing.JLabel tracksFileNameLbl
+LoadFilesDialog(java.awt.Frame parent, boolean modal, MainFrame mainFrame)
-// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initCom
-void planesFileBtnActionPerformed(java.awt.event.ActionEvent evt)
-void tracksFileBtnActionPerformed(java.awt.event.ActionEvent evt)
-void controlStationsFileBtnActionPerformed(java.awt.event.ActionEvent evt)
-void disembarkingStationsFileBtnActionPerformed(java.awt.event.ActionEvent evt)
-void maintenanceStationsFileBtnActionPerformed(java.awt.event.ActionEvent evt)
-void continueBtnActionPerformed(java.awt.event.ActionEvent evt)
MainFrame
-String[] confirmationOptions
-FileHandler fileHandler
-LoadFilesDialog loadFilesDialog
-Simulation currentSimulation
-Thread mfThread
-PlanesInternalFrame planesInternalFrame
-LandingTracksInternalFrame landingTracksInternalFrame
-DisembarkingStationsInternalFrame disembarkingStationsInternalFrame
-MaintenanceStationsInternalFrame maintenanceStationsInternalFrame
-DoubleLinkedList<ControlStationInternalFrame> controlStationsInternalFrame
-javax.swing.JDesktopPane jDesktopPane1
-javax.swing.JLabel jLabel1
-javax.swing.JLabel jLabel12
-javax.swing.JLabel jLabel2
-javax.swing.JLabel jLabel3
-javax.swing.JLabel jLabel4
-javax.swing.JLabel jLabel5
-javax.swing.JLabel jLabel6
-javax.swing.JLabel jLabel7
-javax.swing.JMenu jMenu1
-javax.swing.JMenu jMenu2
-javax.swing.JMenuBar jMenuBar1
-javax.swing.JMenuItem jMenuItem1
-javax.swing.JMenuItem jMenuItem2
-javax.swing.JMenuItem jMenuItem3
-javax.swing.JMenuItem jMenuItem6
-javax.swing.JPopupMenu.Separator jSeparator1
-javax.swing.JInternalFrame welcomeIFrame
+MainFrame()
+void run()
-// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initCom
-void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt)
-void jMenuItem6ActionPerformed(java.awt.event.ActionEvent evt)
-void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt)
-void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt)
+LoadFilesDialog getLoadFilesDialog()
+Simulation getCurrentSimulation()
+void startSimulation()
+void showInternalFrames()
+void updateInternalFramesData()


SetTimesDialog
-int CONSUME_FUEL_TIME
-int LANDING_TIME
-int DISEMBARKING_TIME
-int MAINTENANCE_TIME
-int TAKE_OFF_TIME
-MainFrame mainFrame
-javax.swing.JSpinner consumeFuelSpnr
-javax.swing.JButton continueBtn
-javax.swing.JSpinner disembarkingSpnr
-javax.swing.JLabel jLabel1
-javax.swing.JLabel jLabel2
-javax.swing.JLabel jLabel3
-javax.swing.JLabel jLabel4
-javax.swing.JLabel jLabel5
-javax.swing.JLabel jLabel6
-javax.swing.JSpinner landingSpnr
-javax.swing.JSpinner maintenanceSpnr
-javax.swing.JSpinner takeOffSpnr
+SetTimesDialog(java.awt.Frame parent, boolean modal, MainFrame mainFrame)
-// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents void
-void continueBtnActionPerformed(java.awt.event.ActionEvent evt)
+int getCONSUME_FUEL_TIME()
+int getLANDING_TIME()
+int getDISEMBARKING_TIME()
+int getMAINTENANCE_TIME()
+int getTAKE_OFF_TIME()
LandingTracksInternalFrame
~Simulation currentSimulation
-javax.swing.JButton jButton1
-javax.swing.JLabel jLabel1
-javax.swing.JLabel jLabel2
-javax.swing.JScrollPane jScrollPane1
-javax.swing.JSeparator jSeparator1
-javax.swing.JComboBox<String> landingTrackSelector
-javax.swing.JTable trackPlanesTable
+LandingTracksInternalFrame()
-// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents void
-void landingTrackSelectorActionPerformed(java.awt.event.ActionEvent evt)
-void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
+void updateTracksTable()
+void updateTrackSelector()
+void setCurrentSimulation(Simulation currentSimulation)

PAQUETE: engine


Plane
-int planeId
-String type
-int fuel
-int maxFuel
-int passengers
-Simulation currentSimulation
-ControlStation currentControlStation
-LandingTrack currentLandingTrack
-DisembarkStation currentDisembarkStation
-MaintenanceStation currentMaintenanceStation
-PLANE_STATE planeState
-Plane(int id, String type, int fuel)
+Plane(String[] params)
+void run()
+void fly()
+String[] toTableFormat()
+String[] toAlternateTableFormat()
+void setCurrentSimulation(Simulation currentSimulation)
+int getPlaneId()
+String getType()
+int getFuel()
+PLANE_STATE getPlaneState()
+LandingTrack getCurrentLandingTrack()
+void setPlaneState(PLANE_STATE planeState)
+void setCurrentLandingTrack(LandingTrack currentLandingTrack)
+void setCurrentDisembarkStation(DisembarkStation currentDisembarkStation)
+void setCurrentMaintenanceStation(MaintenanceStation currentMaintenanceStation)
LandingTrack
-int id
-int maxPlanes
-DoubleLinkedList<Plane> planesOnQueue
-Simulation currentSimulation
-Plane currentPlane
-LandingTrack(int id, int maxPlanes)
+LandingTrack(String[] params)
+int getId()
+void setCurrentPlane(Plane currentPlane)
+boolean isTrackFull()
+boolean addLandingPlaneToTrack(Plane plane)
+boolean addTakeOffPlaneToTrack(Plane plane)
+boolean isFirstInLine(Plane plane)
+void freeTrack()
+DoubleLinkedList<Plane> getPlanesOnQueue()
+void setCurrentSimulation(Simulation currentSimulation)
DisembarkStation
-int id
-int maxPlanes
-Simulation currentSimulation
-Plane currentPlane
-DoubleLinkedList<Plane> planesOnDisembarkQueue
-DisembarkStation(int id, int maxPlanes)
+DisembarkStation(String[] params)
+void planeDisembarkRequest(Plane plane)
+void askForDisembark(Plane plane)
+String[] toTableFormat()
+boolean isAvailable()
+int getId()
+void setCurrentPlane(Plane currentPlane)
+void setCurrentSimulation(Simulation currentSimulation)
+DoubleLinkedList<Plane> getPlanesOnDisembarkQueue()

CLASES Y ALGORITMOS - PAQUETE PADRE:

`/com/robertob/p2ipc1/`

PAQUETE: engine

ARCHIVO: `engine/ControlStation.java`

Esta clase representa a las estaciones de control, contienen su ID, la cantidad máxima de aviones a los que pueden comunicarse y la lista de aviones que están actualmente comunicándose. Contiene métodos tanto para establecer y obtener estos valores y también para poder mostrarse en la interfaz gráfica.

ARCHIVO: `engine/DisembarkStation.java`

Esta clase representa a las estaciones de desabordaje, contienen su ID, la cantidad máxima de aviones que pueden tener en cola, el avión que actualmente está desbordando. Contiene métodos tanto para establecer y obtener estos valores y también para poder mostrarse en la interfaz gráfica.

ARCHIVO: `engine/LandingTrack.java`

Esta clase representa a las pistas de aterrizaje, contienen su ID, la cantidad máxima de aviones que pueden tener en cola, el avión que actualmente está utilizando la pista y la lista de aviones que están en cola. Contiene métodos tanto para establecer y obtener estos valores y también para poder mostrarse en la interfaz gráfica.

ARCHIVO: `engine/MaintenanceStation.java`

Esta clase representa a las estaciones de mantenimiento, contienen su ID, la cantidad máxima de aviones que pueden tener en cola, el avión que actualmente está desbordando y la lista de aviones que están en cola. Contiene métodos tanto para establecer y obtener estos valores y también para poder mostrarse en la interfaz gráfica.

ARCHIVO: `engine/Plane.java`

Esta clase representa a un avión, los cuales pueden haber varios en una misma simulación, contienen toda su información, como el ID, la torre a la que se están comunicando, la pista que se les ha asignado desde la torre de control. Contiene métodos tanto para establecer y obtener estos valores y también para poder mostrarse en la interfaz gráfica.

ARCHIVO: `engine/Simulation.java`

Esta clase engloba toda una simulación junto con todos sus elementos, contiene todas las estaciones y aviones, también los tiempos que debe tomar cada acción (configurable) y también es la responsable de inicializar una simulación, junto con todos los hilos y métodos para mostrar información en la interfaz gráfica.

PAQUETE: `frames`

Este conjunto de clases son las responsables de mostrar toda la interfaz gráfica junto con la funcionalidad, apoyándose de las demás clases anteriormente mencionadas, que representan los elementos dentro de una simulación.

ARCHIVO: `frames/ControlStationInternalFrame.java`

ARCHIVO: `frames/DisembarkingStationsInternalFrame.java`

ARCHIVO: `frames/LandingTracksInternalFrame.java`

ARCHIVO: `frames/LoadFilesDialog.java`

ARCHIVO: `frames/MainFrame.java`

ARCHIVO: `frames/MaintenanceStationsInternalFrame.java`

ARCHIVO: `frames/PlanesInternalFrame.java`

ARCHIVO: `frames/SetTimesDialog.java`

PAQUETE: `utils`

ARCHIVO: `utils/list/DoubleLinkedList.java`

Esta clase es una implementación desde cero de una lista doblemente enlazada, la cual es usada en el proyecto para almacenar colecciones de datos u objetos cuando es necesario, salvo algunas excepciones en el caso de la interfaz gráfica como por ejemplo: en el uso del componente *`JTable`*

ARCHIVO: `utils/list/Node.java`

Esta clase es la base de la clase *`DoubleLinkedList`* ya que es la estructura individual la cual implementa la lista doblemente enlazada y es donde se guarda cada dato que sea necesario. Tiene tres parámetros principales, los cuales son el dato que se necesita guardar, una referencia o apuntador al siguiente elemento en la estructura, si aplica, y una referencia o apuntador al previo elemento en la estructura, si aplica.

ARCHIVO: `utils/list/DoubleLinkedListException.java`

Esta clase es una extensión de la implementación de la lista doblemente enlazada y hereda de la clase *`Exception`* que implementa Java para lanzar errores cuando algo no es ejecutado correctamente. Es utilizada cuando se produce algún error en el momento de utilizar y manipular una lista doblemente enlazada.

ARCHIVO: `utils/list/FileHandler.java`

Esta clase es solamente utilizada para la lectura de archivos de entrada que son necesarios para configurar y comenzar una simulación, y tiene relación con la clase *`Simulation`* ya que se necesita de la información dentro de los archivos de entrada para generar elementos para una nueva simulación.