DyanAir agent example

These notes are based on a partial solution developed in the programming "Dojo". See the attached Zip file for Java class files. The files provide enough of a definition to allow the code to run but there is no definition for trains. Your objective is to extend the partial solution contained in the files to include a train class.

The files in the zip file are as follows...

Defs.java
a container for common definitions shared across other classes and defining some aspects of message protocols. This approach simplifies maintenance and allows you to define global variables & utility methods in just one place.

FlightMan.java
this is the main class containing the GUI specification. Read through the FlightMan class definition carefully, it will help you with the other code you need to write. Notice the following...

- I have constructed the primary communicative agents in FlightMan, these agents are then passed as arguments to the constructor methods for Plane, Train & (via Plane) to Passenger;
- there are variables (*noCatches* & *noDrops*) to count the number of passengers who land on the train/ground and JTextFields (*tfCatches* & *tfDrops*) to display these counts;
- a skeleton message listener is provided for the management agent, this should respond to messages from the train about the passengers the train catches or drops;
- there are various bits of code you may choose to amend but I have put comments starting "//##" in places where you will *have* to make changes when you have defined your train class;
- comments starting "/**/" or "//*/" are for enabled/disabled debug statements.

Plane.java
the plane class, it has been amended a little since the *Dojo* event, mostly to simplify the definition and the code-level relationships between planes and other classes. You do not need to edit this class.

Passenger.java
this has also been simplified. You will need to edit this file a little once you have defined Train. "//##" in the Passenger class show the places where you will need to *un*comment calls to sendMessage which passengers use to communicate with trains.

Train.java
this contains a skeleton definition. It is almost completely empty but it provides an appropriate constructor method. You will need to edit this file.

Message Protocol

from manager to plane requesting plane to release a new passenger
- type=drop

from train to manager stating that the train has caught/dropped a passenger
- type=arrival style=catch
- type=arrival style=drop

from passenger to train stating the passenger position as it is falling
- type=falling x=**??** y=**??**

from passenger to train stating the passenger position as it reaches ground level
- type=landed x=**??** y=**??**

Other notes
- to compile and/or run your Java/Boris code you must include boris.jar;
- using ArgList will help you to deconstruct agent messages – see below.

Example Use of ArgList

```
// ArgList helps access data from command strings
// note there should not be spare spaces around '='
String msg = "type=example count=10";

// build an arg list from the string
ArgList mArgs = new ArgList( msg );

mArgs.get("type")      ⇒   "example"
mArgs.get("count")     ⇒   "10"
mArgs.getInt("count")  ⇒   10
mArgs.get("spam")      ⇒   null
```