Conception

There is our java program. It simulates a control tower of an airport. The tower gives us a secured communication system to manage many airplanes.

For the cryptography part, we have done as in the indications on the itp. There is the **KeyPair** class that gives the encrypt and decrypt methods and the **KeyGenerator** Class that generates a new KeyPair.

For messages, we have done it as in the indications too. We create an abstract super class **Message**. So all the messages needed (**Hello**, **SendRSA**, **Keepalive**, **MayDay**, **Bye**, etc.) extend the class Message. All these messages have to redefine the abstract methods sendMessage (used to send the specified message) and accept.(used to allow a MessageHandlerVisitor to visit the specified message).

The **tower** has a **PlaneAccepter** (**ServerSocket**) that waits for a request. When a request arrives (message **Hello**), the **run method** of PlaneAccepter creates a new **Plane** and a new **PlaneHandler**.(socket). There is one PlaneHandler for each Plane. Next, the PlaneHandler answers with a Hello message to the new Plane.

MessageHandler is a class that implements the **Visitor Pattern**. It means that it's the class who manage some messages (Hello, Bye, Choke, UnChoke, Data, MayDay) received by the tower. But when the tower receives a message, she doesn't know what kind of message it is. So the Visitor Pattern is going to check that and do different specified algorithms for each class.

And for the **GUI**, we first create our own, but after we saw that it wasn't much adapted to the situation, so we use circular buffer that was given. For each plane, we create a new thread.

Bonus : We also made Twitter work. It posts a tweet every time a plane lands (The plane sends a Bye message).

Moreover, all the messages that are send are stocked into a **XML file**.