At the airport — 15 points (60 minutes)

a) Create a class `Airplane`, which implements `IPlane`. Implement the methods from the interface following their Javadoc and add necessary member variables and constructors.

b) The interface `IPlane` indirectly also contains the method `compareTo(IPlane)`. It should compare two airplanes according to the following criteria:

- The first criterion is the amount of remaining fuel. The less fuel remains the "smaller" the airplane. I.e. it should be sorted more to the front.
- The second criterion (if the amount of fuel of two airplanes is equal) is the number of passengers. The more passengers are onboard the more to the front the airplane should be sorted.
- The third criterion is the flight code which should be compared ascendingly in lexicographic order.

c) Add sensible implementations of `equals` and `toString` to the `Airplane` class.

d) Create a test class `AirplaneTests` which contains a JUnit test for the `compareTo`-method. The test should check all three of the above criteria.

e) Create a class `Airport` which implements the interface `IAirport`. Implement the methods from the interface following their Javadoc and add necessary member variables and constructors.

The implementation may (but does not need to) look like follows:

- The airport has a list with airplanes in the queue and another list with landed airplanes.
- Airplanes are added/removed from these lists when they approach the airport or are landing.
- The method `getNextPlaneToLand` should return the next ("smallest") airplane in the queue. There are several possibilities to determing the next airplane, e.g. by using `Collections.sort` or a `PriorityQueue`

f) Add a `main` method to the class `Airport` that simulates three airplanes approaching and landing.