Lab # 1 Object Oriented Fundamentals revision with Java

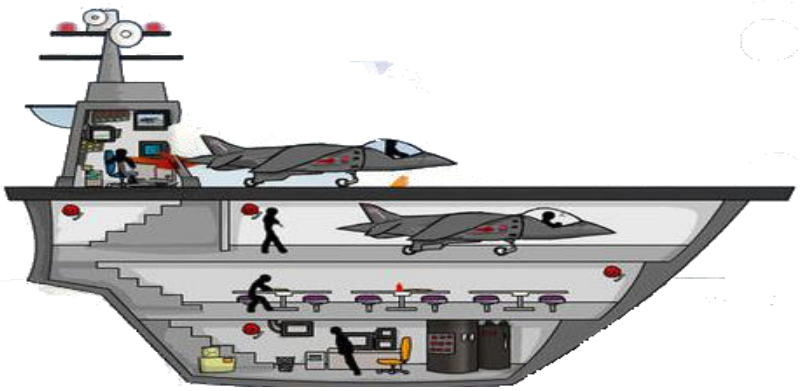
Part A: Setting up Java and necessary files

Make sure you have Java and Eclipse ready. Download Eclipse from eclipse.org if necessary.

Part B: The problem defined

- JUnit test cases for class Aircraft and class Carrier are given. You must import the jar file that contains them by using Import->General->Archive from the menu.
- For how to run JUnit in detail, ask your lecturer or lab assistant.

We are writing a simple simulation of aircraft carriers and aircrafts that can land and take off from the carriers. [Create a project name 2190261_ID_2017 where ID is your id number]



The description of an aircraft and an aircraft carrier are as follows:

An aircraft has the following parameters (**all data must be private**):

- int currentSpeed // current velocity
- int maxSpeed //maximum velocity that the aircraft is capable of

To create an instance of aircraft (class Aircraft) to use in our program we'll need two constructors:

- a no argument constructor: set currentSpeed to 0, maxSpeed to 100.
- A detailed constructor: set currentSpeed and maxSpeed to values that are method arguments. But to prevent an unreasonable value, these setting must be done through method setCurrentSpeed and setMaxSpeed (see their definitions below).

An aircraft has the following operations (**all methods must be public**):

- getCurrentSpeed(): return the value of currentSpeed.
- setCurrentSpeed(int s): set the value of currentSpeed to the value of s. But if s is negative, currentSpeed must be set to 0. If s exceeds maxSpeed, currentSpeed must be set to the value of maxSpeed. This method does not return any value.
- getMaxSpeed(): return the value of maxSpeed.
- setMaxSpeed(int m): set the value of maxSpeed to the value of m. But if m is less than 50, maxSpeed must be set to 50. This method does not return any value. currentSpeed must be adjusted to maxSpeed if it exceeds max speed at this stage.

- toString(): return a string representation of an aircraft. Example: "aircraft: speed= 100, maxSpeed= 200"
- equals(Object o): return true if o is an aircraft with the same currentSpeed and maxSpeed as ***this***.

Write the code for class Aircraft according to the above definition. (8 marks)

Now, we define an aircraft carrier. An aircraft carrier stores the following data (**the data must be private**):

- Aircraft[ ] ac // a storage of aircrafts

To create an instance of an aircraft carrier (class Carrier) to use in our program we'll need two constructors:

- a no argument constructor: set ac to an array that has 5 data slots.
- A detailed constructor: set ac to an array that is a method's argument. All planes in the array must set their speed to zero.

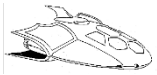
An aircraft carrier has the following operations (**all methods must be public**):

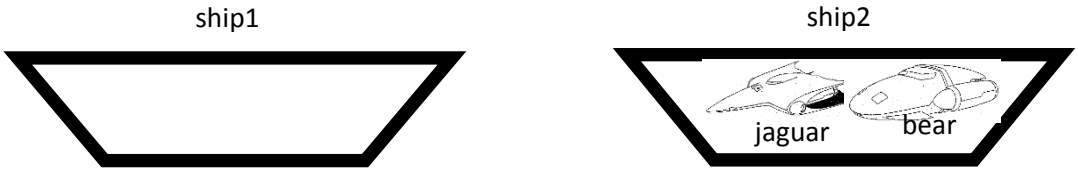
- getCrafts(): return the array, ac, that stores aircrafts.
- planeLand(Aircraft p): store aircraft p in the first empty array slot and set the currentSpeed of p to zero. This method returns true if the aircraft is stored successfully. If the aircraft cannot be stored in the array, this method returns false.
- planeTakeoff(Aircraft p): find aircraft p in the array and remove it from the array (this represents a plane successfully taking off from a standstill position). If the aircraft successfully takes off, set its currentSpeed to 10 and return true. If the aircraft does not successfully take off (maybe it is not stored in the array), just return false.
- toString(): return a string representing all aircrafts stored inside. Example (if there are 2 aircrafts):
  "aircraft: speed= 0, maxspeed= 200
  aircraft: speed= 0, maxspeed= 230"

Write the code for class Carrier defined above (5 marks).

A simulation program of carrier and aircrafts is to be written in class MyProgram. The initial state of the program world is:

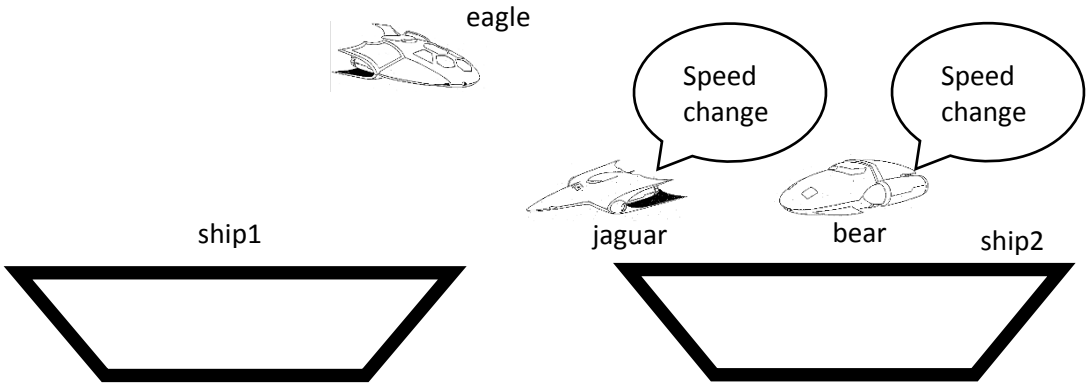
- ship1 is a carrier. No aircraft is stored inside. It can store 5 aircrafts.
- ship2 is a carrier that has 2 aircrafts inside:
  - jaguar – a standard aircraft.
  - bear – a standard aircraft.
- eagle – a flying aircraft, with speed = 50 and maxSpeed = 100.





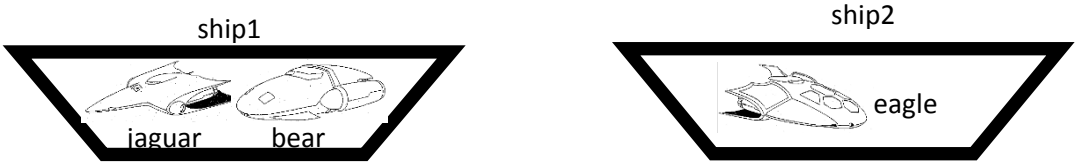
We want to simulate a two-state change in this simulation. The first state happens when:

- jaguar and bear leave ship2.
- The jaguar pilot then attempts to change jaguar's speed to 105.
- The bear pilot then attempts to change bear's speed to 90.



The second state happens when:

- ship1 receives the landed jaguar and bear.
- ship2 receives eagle.



Write the main method of class MyProgram, simulating each state (***6 marks***).

A starting template is given below. There is no JUnit test for this main program. Write tests by yourself (in the main method or as your own JUnit) if you need them.

```
public class MyProgram {
     public static void main(String[] args) {
          // initial

          // state change 1

          // state change 2

     }
}
```

Create a jar file 2190261_ID_2017 with all your source code (including the tests) and submit the jar file through courseville.