*<<interface>>*

*Flier*

*fly( )*

*Fly( )*

*<<interface>>*

*Athlete*

*fly( );*

*train()*

*Fly( )*

*Bird*

*fly( )*

*AirePlane*

*fly( )*

*SkiJumper*

*fly( )*

## PART 1

Implement all classes/interfaces as defined in the above UM Class diagram. The fly method for each of the concrete classes shall implement flying in its own way. Write **a test class** that create an ArraryList (only one array) of objects of each type (Bird, AirePlane and SkiJumper) and ploymorphically Iterate through all elements of the array and invoke the fly method.

Notes:

* Simulate flying in any way you like (print a message indicating how you fly for example).
* A SkiJumper is an Athlete who trains for hours in a certain sport. And the class shall also contain information about the athlete (name, number of jumps, hours training, etc).
* Your classes shall be fully encapsulated.

## PART 2

Add another interface to the design and call it ***Displayable*** with one method called **display().** Since you are using Java 8 add a default implementation to the display method and have all other classes implement the new interface with the exception of the SkiJumper as it will use the default implementation.

## PART 3 (to be completed only after you fully test all classes in PART1):

Add 2 more flier athlete objects to your design and write a short summary of what exactly did you have to change indicating whether the changes were invasive or not?

I added two classes names SkyDiver and Astronaut, which inherited from Athlete and Flier. In each of these classes, I gave them their own implementation of the fly and train method. These changes were not invasive, as I only modify within the classes themselves, and the interface allows the user to know which methods they can call from each of the classes.