Lab Interface

- Download LabInterface.zip from Canvas and extract the files (unzip it).
- Import the code into Eclipse:
  - In Package Explorer (Eclipse) right-click the folder that includes all your labs > **Import** .. the Import dialog opens
  - Select ***General*** > ***File System*** and click ***Next*** the Import from directory dialog opens
  - User the ***Browse..*** button to navigate to the directory that includes the extracted files
  - Check the folder on the left panel (*labInterface*). All files on the right will be checked
  - Ensure that the *Create top-level folder* checkbox is checked. Then click ***Finish***
- Run *App* to make sure that the file import worked as expected.

- Create a new interface called Flyable
  - Right-click labInterface > **New** > **Interface**
- In the new interface declare two methods called launch and land
  Both have a return type void, both have 0 parameters.
  Remember: methods declared in an interface are implicitly public and abstract.
  They need no access-modifier and they have no method body but a semi-colon after the method header.
- Change Bird, Hangglider, and Plane so that all three classes implement Flyable.
  - A wiggly line will appear under the class name
  - Roll your mouse over the underlined class name and select ***Add unimplemented methods***
    Eclipse will create the appropriate method stubs for you
- Implement the interface methods by printing the following messages – each terminated with a new line
  - Bird launch: Flapping the wings to take off
    Bird land: Flapping the wings until landing
  - Hangglider launch: Running until take-off
    Hangglider land: Gliding to a land
  - Plane launch: Rolling until take-off
    Plane lane : Rolling to a stop
- In App add an empty line to structure the output
- Then create an array of Flyable and name it flyingThings
  Use an array initializer to initialize it with myPlane, myHangglider, and myBird
- Use a foreach loop to loop through all the elements of flyingThings
  For each of the elements print the element and call both launch and land.
  Structure your output with single empty lines.

THE END

OUTPUT:

```
myPlane: Boing 747 with 4 engines
myHangglider: Rigid-wing
myBird: Swallow

Boing 747 with 4 engines
Rolling until take-off
Rolling to a stop

Rigid-wing
Running until take-off
Gliding to a land

Swallow
Flapping the wings to take off
Flapping the wings until landing
```