Instructions for exercise 6
This exercise closes at 3pm Thursday March 10th 2022 in Brightspace (except for students with special accommodations)

The Mediator pattern is about **localizing complex interactions** between objects into **one** method of the mediator. This pattern rests on the idea that the mediator has one and only one method (WidgetChanged in the example of the slides) to handle a change from any of the 'colleagues' (i.e., 'mediated' or equivalently 'coordinated' instances; in the example, the Widgets). That allows all consequences of adding, modifying or deleting coordinated instances and their interactions to be localized in a single method. The negative side of the pattern is that this method tends to grow quickly.

The Composite pattern can be summarized as follows: make a collection of instances of a class X **look and behave as** though this is not a collection but **a single instance** of class X.

You are assumed to have studied both patterns.

This exercise consists of 3 steps. **All code must be in the different branches of a new repo.**

In step 1, you are to implement the mediator Go4 pattern as described above.
In step 2, the mediator of step1 is augmented with the ability to use a composite object.

Step 3 is separate from the first two steps. It explores a mediator that defines interactions with respect to the class of the different coordinated instances.

As in all pattern-based exercises of this term, you are marked on your ability

1) to obtain code that runs correctly AND
2) to follow the organization a particular pattern 'imposes' to a solution.

Code that does not run gets a mark of 0.
Code that does run is then evaluated with respect to whether or not it respects the required pattern(s).

Assigned Tas (to invite to your repo for ex6)

From Abbasi to Djani:	Anant (anantojha)
From Edwards to Kebedom:	Alexei (alexeikrumshyn)
From Kim to Peckham:	Projna (projna)
From Po to Zhu: Samin:	Samin (Samin005)

Step 1. (Static Mediator)
Consider the files mainStatic.java and StaticMediator.java.
You cannot modify mainStatic.java
For step1, consider method *runScenario1* found in StaticMediator.java
Create in IntelliJ a project called ex6Static and add the missing classes and methods in order to have the mainStatic generate the output found in ex61.rtf

In particular, the following interactions must be correctly handled by StaticMediator:
When s1 changes status then s1 and s2 swap their secret
When s2 changes status then s3's secret is appended to the end of s2's secret
When s3 changes status then s3's clearance is to the clearance of s1 + the clearance of s2
And the clearance of s4 is set to 0
When s4 changes status then
If s4's clearance is > 0 then s1's clearance is set to s4's clearance
And s2's clearance is set to s3's clearance
Else do nothing
When a1 changes status then a1's secret is set to s3's secret
And the secret of m1 is set to "forgotten"
When m1 changes status m1 and s4 swap secrets
When c1 changes status a1 and m1 are set to null

Once your code is correctly organized and generates the required output, submit it to a repository created for ex6. Do invite your designated TA to that repo. ALSO submit your zipped source for step1 (called step1) AND a link to your repo (in a .txt file) to Brightspace.

Step 2 (Composite)
In mainStatic, comment out line 5 (which calls runScenario1) and uncomment line 7 that calls runScenario2.
In StaticMediator, uncomment the lines preceded by the comment //step 2

Add the following interactions to StaticMediator:
When a2 changes status then set its secret to a1's secret followed by m1's secret
When m2 changes status then set secret of s1 to m2's secret
When a3 changes status then

- set duo's secret (ie the secret of its each of its components) to the secret of a1 followed by the secret of m2
- set the secret of a3 to the secret of s1
- set the secret of m3 to "forgotten"

When m3 changes status then swap the secret of duo with the secret of m1

Add whatever is necessary to your project so that your solution:

1) Produces the correct output (see ex62.rtf)
2) Obeys the code organization of the Composite pattern for class CompositeAgent

Once your code for step 2 is correctly organized and generates the required output, submit it to a new branch of your repository for ex6. Also submit your zipped source for step 2 (called step2) to Brightspace.

Step 3 (Dynamic Mediator)
Create a new branch in your repo for step3.
Create a new IntelliJproject called ex6ClassBased with the main program given in file mainClassBased.java.
I also supply you with an incomplete class Mediator.java

Class Mediator must encode the following interactions:
1) When any instance of class Agent changes status
  Then set the secret of that Agent to the concatenation of the secrets of all current moles
  And printout the name that agent and it's new secret
2) When any instance of class Moles changes status
  Then the secret of this mole is appended to the end of the secret of the first agent added to the team and the new secret of that first agent is output
  And then the secret of that mole is set to "mothing to say " and output
3) When any instance of class Cleaner changes status
  Then all current moles are set to null (i.e., eliminated) and the messages "moles have been cleaned up" is output

Add whatever is necessary to your project so that your solution:

1) Produces the correct output (see ex63.rtf)
2) Obeys the code organization of the Go4 Mediator pattern

Once your code for step 3 is correctly organized and generates the required output, submit it to a new branch of your repository for ex6. Also submit your zipped source for step 3 (called step3) to Brightspace.