C5- S5 – PRACTICE

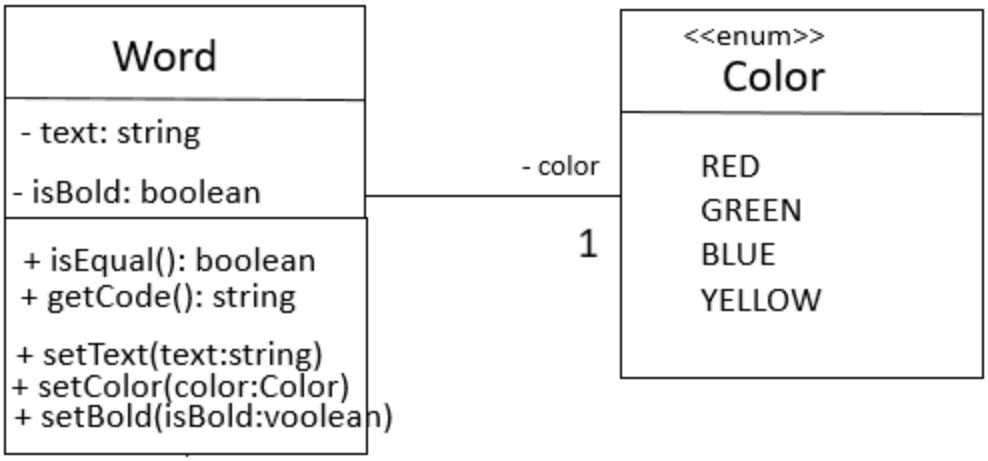
⚠ Your project must include a `tsconfig.json` file and build JS files in `/dist` folder

⚠ Each class must be in a separate file *(example: Rectangle.ts)*

⚠ You also need to create a Main.ts file to test all your shapes

We want to manage the edition of a world

- Set the world text
- Set the world bold
- Set the world color



Example: new World ('ronan', true, Color.GREEN0 should produce: **ronan**

Q1:

- Write the code corresponding to the UML diagram
- Implement the method isEqual to test if 2 words are equal
- Also implement the setters

Q2:

- Write the method getCode(), which returns an HTML-based code to describe the word

Example: new World ('ronan', true, Color.GREEN) should produce:

```
<word   text='ronan', bold='false', color='red' />
```

The model of actions

We want to **undo actions performed on a word**

Example of scenario:

#	ACTION	TEXT VALUE	COMMENT
1.	Set text to “RONAN”	RONAN	
2.	Set text to bold	**RONAN**	
3.	Set color “red	**RONAN**	
4.	Set text to “RADY”	**RADY**	
5.	Undo	**RONAN**	We undo the set text to rady
6.	Undo	**RONAN**	We undo the set color to red
7.	Undo	RONAN	We redo the set text to bold

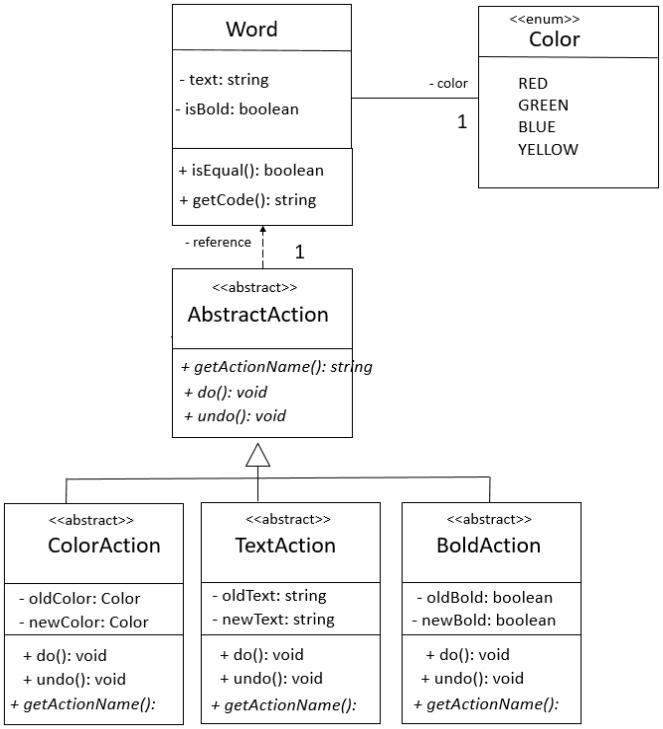
To record the actions we first need to implement **the model of actions:**

- AbstractAction is an abstract class composed of:
  - The **reference** to the word object
  - The abstract methods : actionName(), undo() , do()

We define **3 concrete actions**: ColorAction, BoldAction and TextAction

- Each children class must implement the abstract methods:
  - **actionName** is the name of the action (hard coded in the children class)
  - **undo**() is called when action need to be un-done : the previous value need to be set.
  - **do**() is do when action need to be done (*or re-done but this is for this BONUS part)* : the new value needs to be set.

- Each action contains its specific old and new values



Q3:

- Implement the model of actions following the UML diagram
- Implement the do/undo method on each concrete classes : they must set the old or the previous attribute to the reference word

The history of actions

Now we define the container of actions: the ActionHistory

- To perform an action on the word, the method **do(action)** must be called on ActionHistory

  Example:

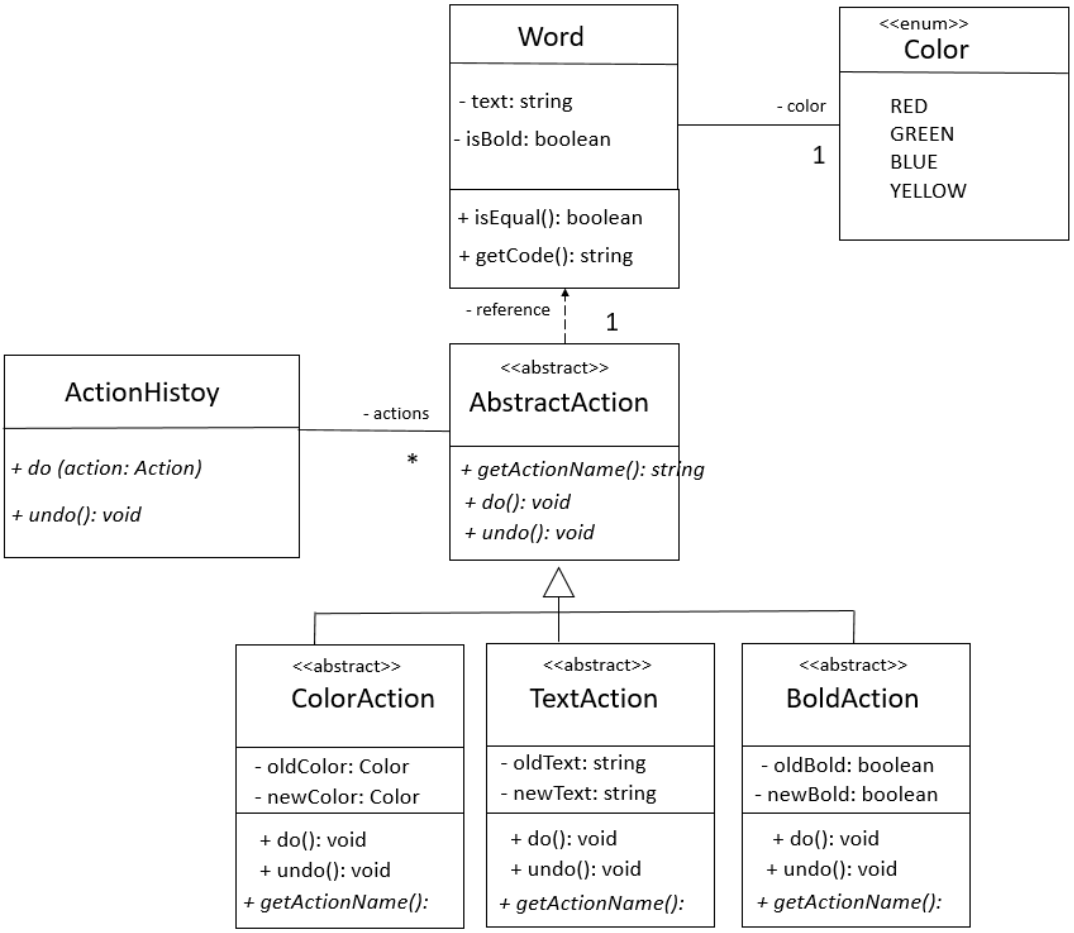
```
actionHistory.do(new ColorAction(theWorld, Color.RED));
```

<u>The do() method of ActionHistory will :</u>

1. Call the do() method of the action
2. **Push** the action to the list of action

<u>The undo() method of ActionHistory will :</u>

1. **Pop** the last action from the list of actions
2. Call the undo() method on this action



Q4:

- Implement the ActionHistory class and the undo/redo mechanism
- To test your application you need to LOG every action :
    - Each action need to log on the console their change
        - Example :
            - set RED color
            - restore BLUE color

- Here is a MAIN to test your application:

```
let actionHistory = new ActionHistory();
let ronanWord = new World (‘RONAN, true, Color.GREEN);

actionHistory.do(new ColorAction(theWorld, theWorld.getColor(),Color.RED));
actionHistory.do(new ColorAction(theWorld, Color.BLUE));
actionHistory.do(new TextAction(theWorld, theWorld.getText(), "RADY"));
actionHistory.undo();
actionHistory.undo();
actionHistory.undo();
```

- The program shall print

```
set RED color
set BLUE color
set text RADY
```

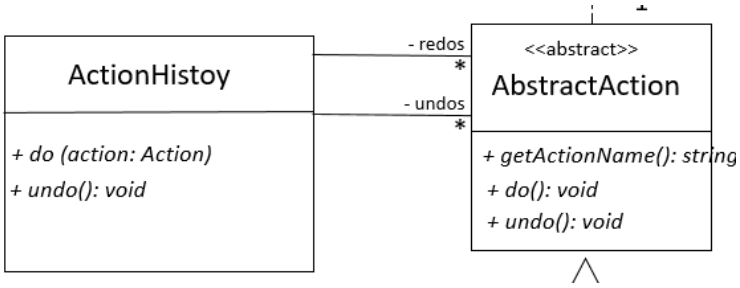
```
restore RONAN text
restore RED color
restore GREEN color
```

- And the word should be and the end : **RONAN**

BONUS for crazy coders

We would like also to REDO actions

- To do so, we need now 2 list of actions :
  - the undos (the list of action to be undone)
  - the redos (the list of actions to be re-done)



Whenever an action is UNDONE, the action move to the list of REDO actions

Q5: Find the way to implement this new REDO feature and update the ActionHistory class