

REQ6: Monologue

Overview:

To achieve this feature, there will be two new classes (i.e., SpeakAction and Toad) created in the extended system, and two existing classes (i.e., Application and Status) will be modified. The design rationale for each new or modified class is shown on the following pages.

1) SpeakAction

What changed in the design rationale between Assignment 1 and Assignment 2 and Why:

There is no changes in this class of REQ6 between Assignment 1 and Assignment 2.

2) Toad

What changed in the design rationale between Assignment 1 and Assignment 2 and Why:

In Assignment 2, I add one method `getReplyString()` to modify which sentence to be talk depends on different capabilities the player had. In Assignment one, there are three methods which have the similar code structure, which are: `giveRandomTalk()`, `noTalkPowerStar()` and `noTalkWrench()`. The only difference between them is the index of the sentence to be remove. It does not obey the design principle DRY, so I add one `getReplyString()` method for this class, to make it more logical and readable.

```
public String getReplyString(Integer removeIndex){
```

Why I choose to do it that way:

To reduce the repetitive code in those method, I add this method which has an index called `removeIndex` (represent the index will be remove) as its input parameter and return an `currentIndex`(after fulfil some conditions). In this method, if the `removeIndex` equals to null, then the `currentIndex` will be any random number within the size of the `toadTalk` Array.

Else we put the `currentIndex` as any random number within the size of the `toadTalk` Array. If the `currentIndex` equals to the `removeIndex`, the `currentIndex` will jump over that `removeIndex`, so the corresponding sentence will not be print.

This method helps to get the correct sentences, so when I want to print the corresponding sentences, I just call this method and input the index of the sentence which I want to remove from the arraylist. It obeys the design principle and make the code logical.

```
public String giveRandomTalk() { return getReplyString( removeIndex: null); }  
public String noTalkPowerStar() { return getReplyString(POWER_STAR_INDEX); }  
public String noTalkWrench() { return getReplyString(WRENCH_INDEX); }
```

3) Application

What changed in the design rationale between Assignment 1 and Assignment 2 and Why:

There is no changes in this class of REQ6 between Assignment 1 and Assignment 2.

4) Status Enum

What changed in the design rationale between Assignment 1 and Assignment 2 and Why:

In Assignment 2, I change the capability name after player consume the power star, instead of using POWER_STAR_BUFF, I will use INVINCIBLE. POWER_STAR_BUFF is not so readable as INVINCIBLE.

```
INVINCIBLE, // use this status to determine player has consumed the Power star
```

Why I choose to do it that way:

In the assignment specification, it mentions that after player consume the power star, he will become invincible, so I think INVINCIBLE will be more suitable for this capability. It helps me to give capability to player after player consume a power star. By using this enum class, it provides us more flexibility for future development as we can simply add additional capability when needed.