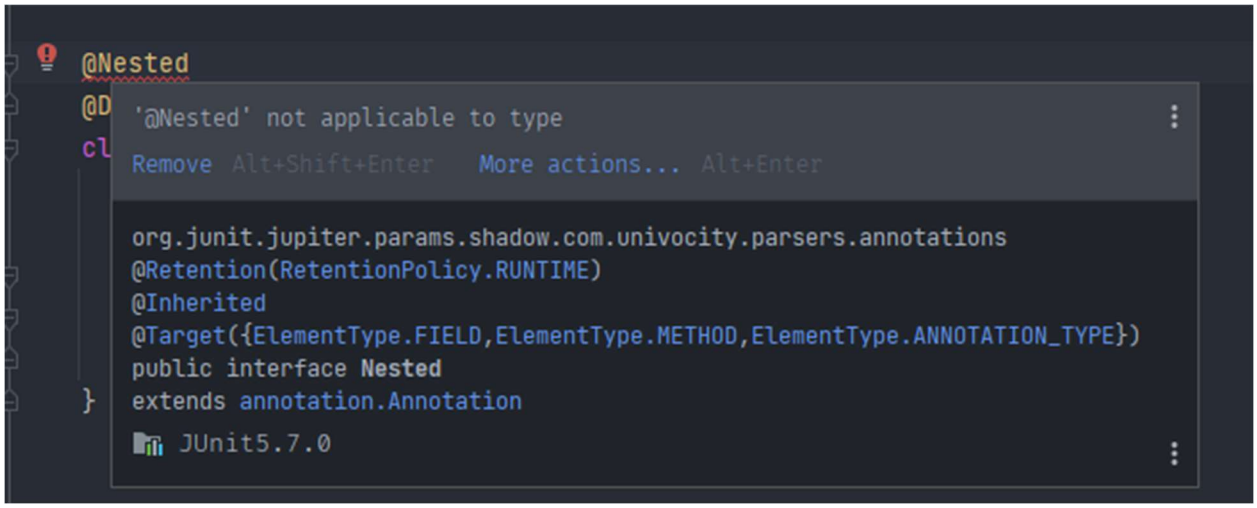
Testing

- I wrote a function to be run before each test functions.
- Tried to use nested test but faced an error.



- Tried to use conditional test execution but couldn't figure out how to write custom conditions.
- I wrote tests to check:
  - If name given is valid.
  - If quality given is valid.
  - Test quality of Aged Brie.
  - Test quality of Backstage pass for:
    - Before 10 days or after sellIn.
    - Within 10 to 6 days before sellIn.
    - 5 to 1 day before sellIn.
  - Test quality of Sulfuras.
  - Test any other ordinary items.

Refactoring

Iteration 1:

- Used functional programming paradigm to create separate function for each uses.
    - A function for Aged Brie.
    - A function for Backstage Pass.
        - A function to calculate each scenario of Backstage Pass.
    - A function for Sulfuras
    - A function for ordinary items before sellIn date passes.
    - A function for ordinary items after sellIn date passes.

Added two videos. One before refactoring and one after refactoring.

Iterations:
Iteration – 1:

```
//Iteration 2
class Inventory {
    Item[] items;

    public Inventory(Item[] items) {
        this.items = items;
    }

    public void updateQuality() {
        for (int i = 0; i < items.length; i++) {
            if (!items[i].name.equals("Aged Brie")
                    && !items[i].name.equals("Backstage passes t
o a TAFKAL80ETC concert")) {
                if (items[i].quality > 0) {
```

```
                    if (!items[i].name.equals("Sulfuras, Hand of
 Ragnaros")) {
                        items[i].quality = items[i].quality - 1;
                    }
                }
            } else {
                if (items[i].quality < 50) {
                    items[i].quality = items[i].quality + 1;

                    if (items[i].name.equals("Backstage passes t
o a TAFKAL80ETC concert")) {
                        if (items[i].sellIn < 11) {
                            if (items[i].quality < 50) {
                                items[i].quality = items[i].qual
ity + 1;
                            }
                        }

                        if (items[i].sellIn < 6) {
                            if (items[i].quality < 50) {
                                items[i].quality = items[i].qual
ity + 1;
                            }
                        }
                    }
                }
            }

            if (!items[i].name.equals("Sulfuras, Hand of Ragnaro
s")) {
                items[i].sellIn = items[i].sellIn - 1;
            }

            if (items[i].sellIn < 0) {
                if (!items[i].name.equals("Aged Brie")) {
                    if (!items[i].name.equals("Backstage passes
to a TAFKAL80ETC concert")) {
                        if (items[i].quality > 0) {
```

```
                            if (!items[i].name.equals("Sulfuras,
 Hand of Ragnaros")) {
                                items[i].quality = items[i].qual
ity - 1;
                            }
                        }
                    } else {
                        items[i].quality = items[i].quality - it
ems[i].quality;
                    }
                } else {
                    if (items[i].quality < 50) {
                        items[i].quality = items[i].quality + 1;
                    }
                }
            }
        }
    }
}
```

Iteration – 2:

```
//Iteration 2
class Inventory {
    Item[] items;
    int i=0;

    public Inventory(Item[] items) { this.items = items; }

    public int updateQualityAgedBrie(Item item) {
        if(item.name.equals("Aged Brie") && item.quality <50)
        {
            //System.out.println("Aged Brie.");

            ++item.quality;
        }

        return item.quality;
```

```
    }

    public int updateQualityBackstagePassWithinLimit(Item item,
int higherLimit) {
        if(item.name.equals("Backstage passes to a TAFKAL80ETC c
oncert") && item.quality <50 && item.sellIn > 0 && item.sellIn <
 higherLimit)
        {
            ++item.quality;

            //System.out.println("Backstage pass" + 0 + " " + hig
herLimit);
            //System.out.println(item.quality);
        }

        return item.quality;
    }

    public int updateQualityBackstagePassBeforeSellIn(Item item)
 {
        if(item.name.equals("Backstage Pass") && item.quality <5
0 && item.sellIn > 0)
        {
            ++item.quality;
        }

        return item.quality;
    }

    public int updateQualityBackstagePass(Item item) {
        if(item.name.equals("Backstage passes to a TAFKAL80ETC c
oncert") && item.quality <50 && item.sellIn <= 0)
        {
            return 0;
        }
        item.quality = updateQualityBackstagePassWithinLimit(ite
m,11);
        item.quality = updateQualityBackstagePassWithinLimit(ite
m,6);
```

```
        item.quality = updateQualityBackstagePassWithinLimit(ite
m, Integer.MAX_VALUE);

        return item.quality;
    }


    public int updateQualitySulfuras(Item item) {
        if(item.name.equals("Sulfuras, Hand of Ragnaros"))
        {
            //System.out.println("Sulfuras");
            item.quality = 80;
        }

        return item.quality;
    }

    public int updateQualityOrdinaryItemsBeforeSellIn(Item item)
 {
        if(!item.name.equals("Sulfuras, Hand of Ragnaros") && !i
tem.name.equals("Backstage passes to a TAFKAL80ETC concert")
                && !item.name.equals("Aged Brie") && item.qualit
y > 0 && item.sellIn > 0)
        {
            //System.out.println("Ordinary item found.");
            --item.quality;
        }

        return item.quality;
    }

    public int updateQualityOrdinaryItemsAfterSellIn(Item item)
{
        if(!item.name.equals("Sulfuras, Hand of Ragnaros") && !i
tem.name.equals("Backstage passes to a TAFKAL80ETC concert")
                && !item.name.equals("Aged Brie") && item.qualit
y > 0 && item.sellIn <=0)
        {
            //System.out.println("Ordinary item found.");
```

```
            item.quality -= 2;
        }

        return item.quality;
    }

    //After refactoring

    public void updateQuality() {   //this method updates the qua
lity for one day
        for (Item item : items) {
            //System.out.println(i);

            item.quality = updateQualityAgedBrie(item);
            item.quality = updateQualityBackstagePass(item);
            item.quality = updateQualitySulfuras(item);
            item.quality = updateQualityOrdinaryItemsBeforeSellI
n(item);
            item.quality = updateQualityOrdinaryItemsAfterSellIn
(item);

            //i++;
        }




    }
}
```