UNIVERSIDAD NACIONAL DE COLOMBIA

PERFICIENT

Homework #2:

Good practices

Made by:

Sebastián Giraldo Villegas

Paula Andrea Taborda Jaramillo

Professor:

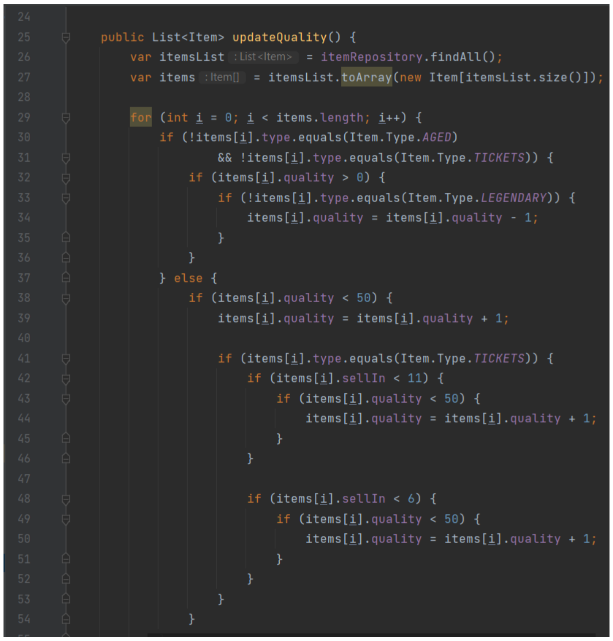
Cesar Alarcón Manrique / Luis Paz Velásquez

Praxis 2022-1

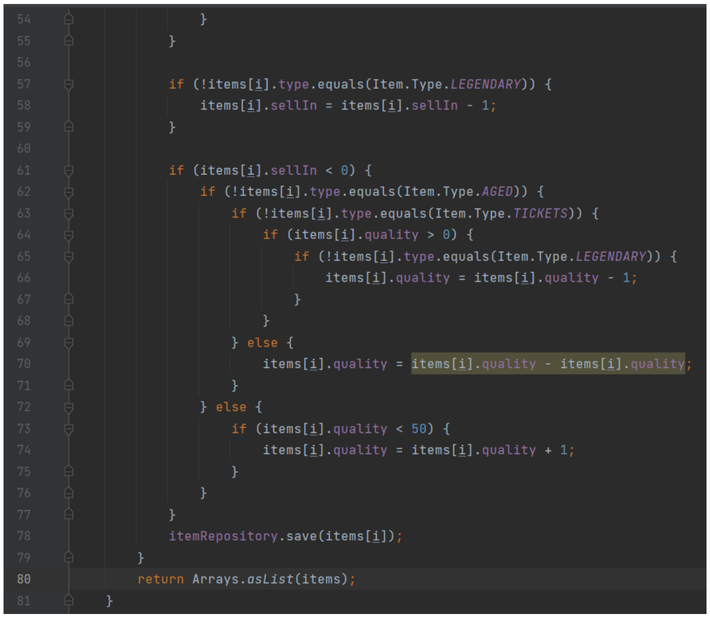
Medellín

Fixing the code:

1. Our first step for cleaning the code was analyzing how updateQuality() was working. Also we read the Read.me, in that way we could understand what the code was doing.



```
    public List<Item> updateQuality() {
        var itemsList :List<Item> = itemRepository.findAll();
        var items :Item[] = itemsList.toArray(new Item[itemsList.size()]);

        for (int i = 0; i < items.length; i++) {
            if (!items[i].type.equals(Item.Type.AGED)
                    && !items[i].type.equals(Item.Type.TICKETS)) {
                if (items[i].quality > 0) {
                    if (!items[i].type.equals(Item.Type.LEGENDARY)) {
                        items[i].quality = items[i].quality - 1;
                    }
                }
            } else {
                if (items[i].quality < 50) {
                    items[i].quality = items[i].quality + 1;

                    if (items[i].type.equals(Item.Type.TICKETS)) {
                        if (items[i].sellIn < 11) {
                            if (items[i].quality < 50) {
                                items[i].quality = items[i].quality + 1;
                            }
                        }

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

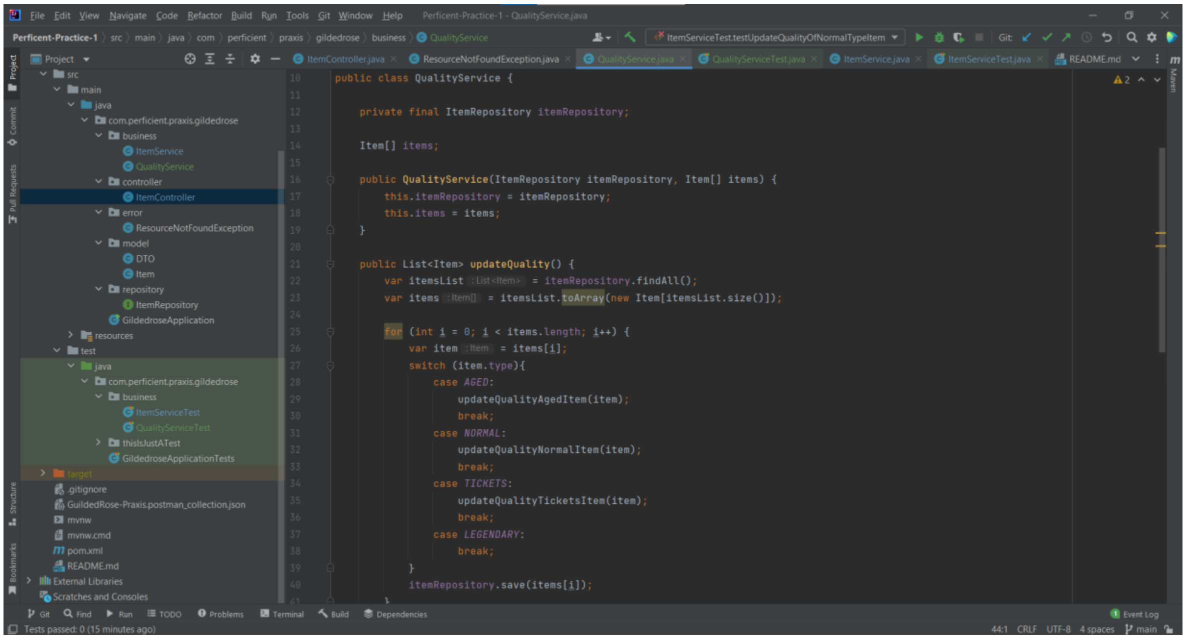


```
                }
            }

            if (!items[i].type.equals(Item.Type.LEGENDARY)) {
                items[i].sellIn = items[i].sellIn - 1;
            }

            if (items[i].sellIn < 0) {
                if (!items[i].type.equals(Item.Type.AGED)) {
                    if (!items[i].type.equals(Item.Type.TICKETS)) {
                        if (items[i].quality > 0) {
                            if (!items[i].type.equals(Item.Type.LEGENDARY)) {
                                items[i].quality = items[i].quality - 1;
                            }
                        }
                    } else {
                        items[i].quality = items[i].quality - items[i].quality;
                    }
                } else {
                    if (items[i].quality < 50) {
                        items[i].quality = items[i].quality + 1;
                    }
                }
            }
            itemRepository.save(items[i]);
        }
        return Arrays.asList(items);
    }
```

1. After that, we translated the complicated and unnecessarily code to a new one and we proved that it was working before making other changes. We tried to loose coupling in every step.

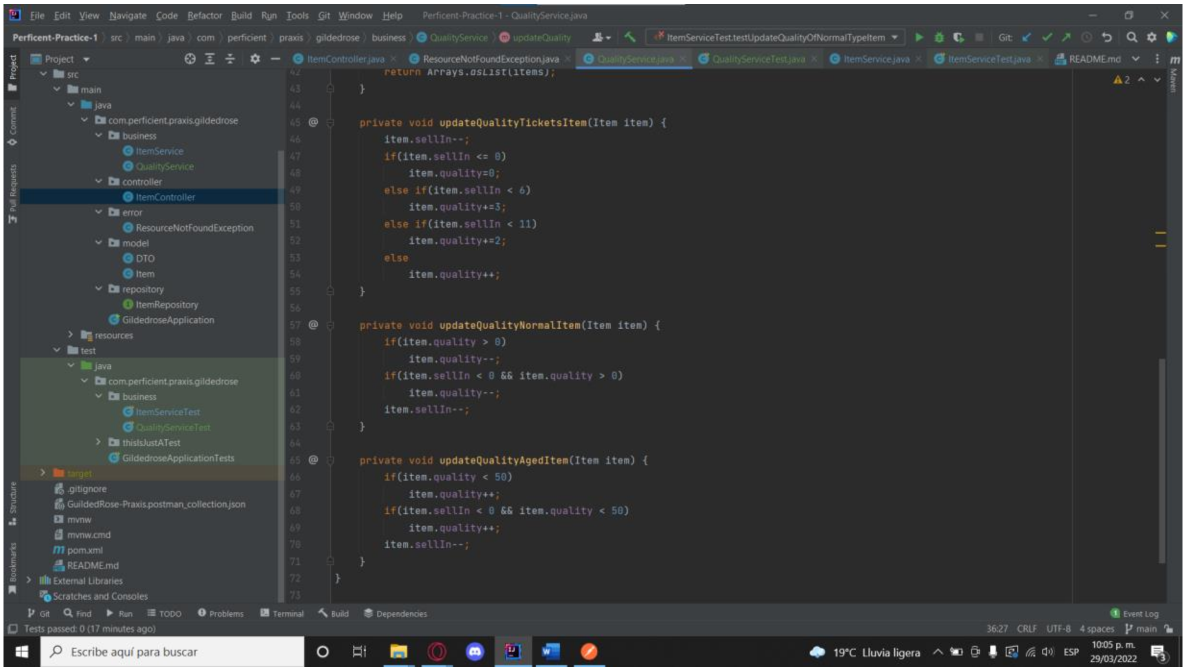
```
var item = items[i];
            switch (item.type){
                case AGED:
                    if(item.quality < 50)
                        item.quality++;
                    if(item.sellIn < 0 && item.quality < 50)
                        item.quality++;
                    item.sellIn--;
                break;
                case NORMAL:
                    if(item.quality > 0)
                        item.quality--;
                    if(item.sellIn < 0 && item.quality > 0)
                        item.quality--;
                    item.sellIn--;
                break;
                case TICKETS:
                    if(item.sellIn <= 0)
                        item.quality=0;
                    else if(item.sellIn < 6)
                        item.quality+=3;
                    else if(item.sellIn < 11)
                        item.quality+=2;
                    else
                        item.quality++;
                break;
                case LEGENDARY:
                    break;
```

3. Then, we created a class called QualityService where we passed all the content from the updateQuality() method that was in ItemService.


Perficent-Practice-1 - QualityService.java
ItemController.java
ResourceNotFoundException.java
QualityService.java
QualityServiceTest.java
ItemService.java
ItemServiceTest.java
README.md
ItemServiceTest.testUpdateQualityOfNormalTypeItem
Tests passed: 0 (15 minutes ago)

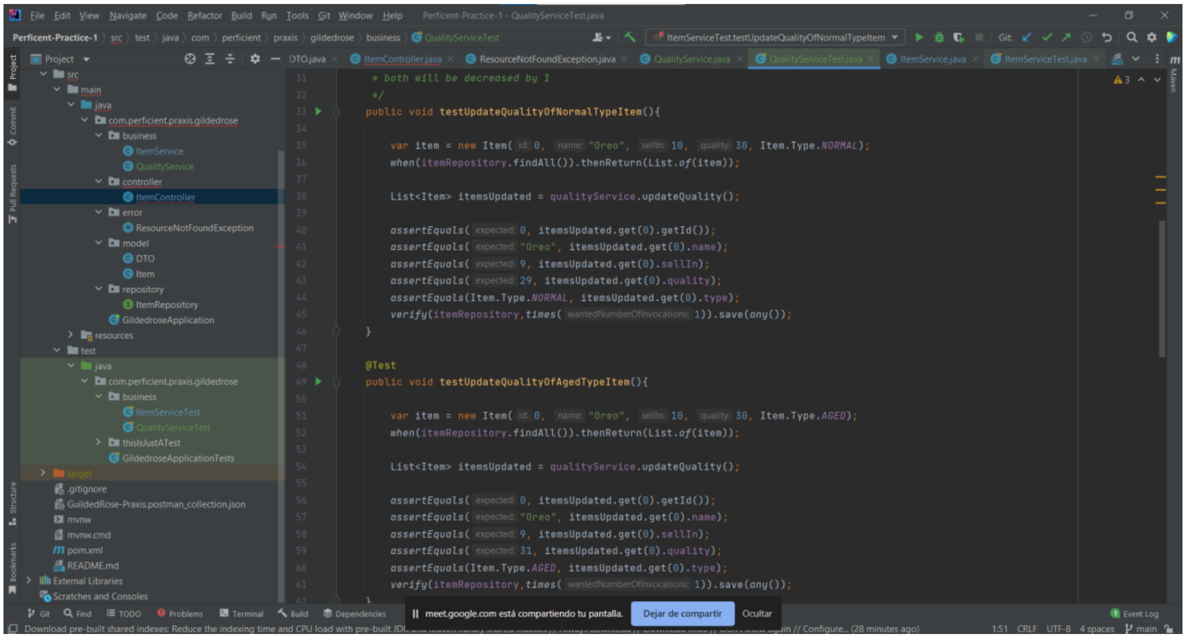
```java
public class QualityService {

    private final ItemRepository itemRepository;

    Item[] items;

    public QualityService(ItemRepository itemRepository, Item[] items) {
        this.itemRepository = itemRepository;
        this.items = items;
    }

    public List<Item> updateQuality() {
        var itemsList = itemRepository.findAll();
        var items = itemsList.toArray(new Item[itemsList.size()]);

        for (int i = 0; i < items.length; i++) {
            var item = items[i];
            switch (item.type){
                case AGED:
                    updateQualityAgedItem(item);
                    break;
                case NORMAL:
                    updateQualityNormalItem(item);
                    break;
                case TICKETS:
                    updateQualityTicketsItem(item);
                    break;
                case LEGENDARY:
                    break;
            }
            itemRepository.save(items[i]);
        }
```

4. Inside the class QualityService we created the new methods for every type of item, with the exception of Legendary, because it should not be touched.


Perficent-Practice-1 - QualityService.java
QualityService
updateQuality
ItemServiceTest.testUpdateQualityOfNormalTypeItem
Tests passed: 0 (17 minutes ago)
Escribe aquí para buscar
19°C Lluvia ligera
ESP
10:05 p. m.
29/03/2022

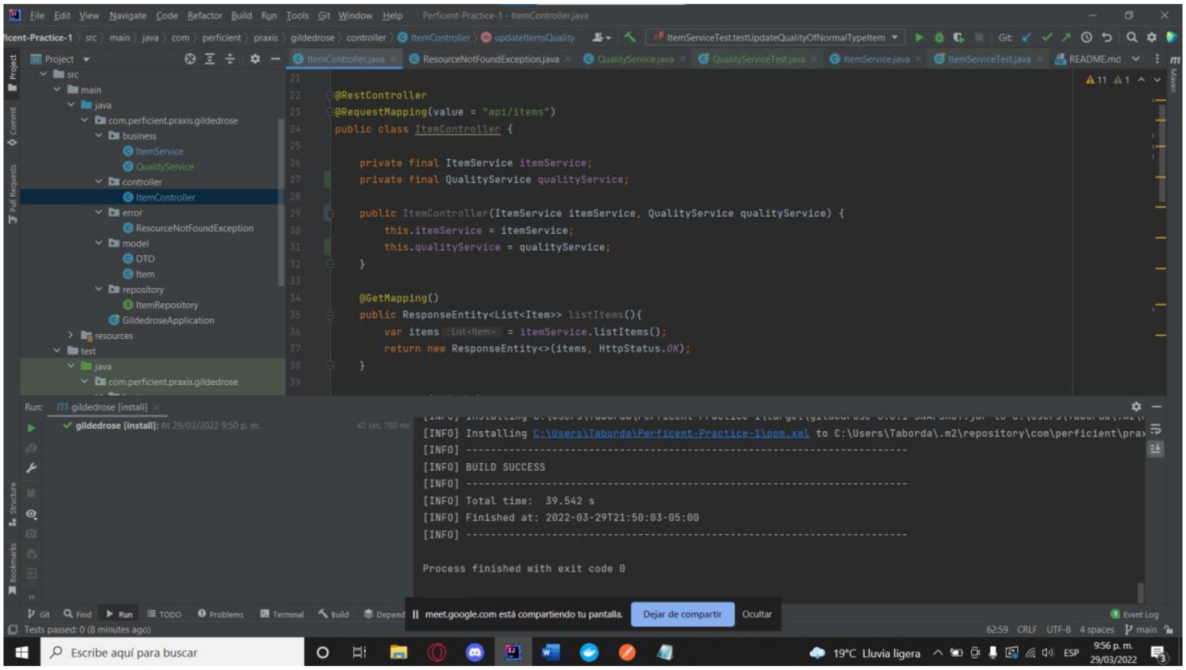
```java
        return Arrays.asList(items);
    }

    private void updateQualityTicketsItem(Item item) {
        item.sellIn--;
        if(item.sellIn <= 0)
            item.quality=0;
        else if(item.sellIn < 6)
            item.quality+=3;
        else if(item.sellIn < 11)
            item.quality+=2;
        else
            item.quality++;
    }

    private void updateQualityNormalItem(Item item) {
        if(item.quality > 0)
            item.quality--;
        if(item.sellIn < 0 && item.quality > 0)
            item.quality--;
        item.sellIn--;
    }

    private void updateQualityAgedItem(Item item) {
        if(item.quality < 50)
            item.quality++;
        if(item.sellIn < 0 && item.quality < 50)
            item.quality++;
        item.sellIn--;
    }
}
```

5. Our next step was creating a new class for the tests associated to QualityService, then we wrote there the ones that were previously in ItemService.



```java
     * both will be decreased by 1
     */
    public void testUpdateQualityOfNormalTypeItem(){

        var item = new Item( id: 0,  name: "Oreo",  sellIn: 10,  quality: 30, Item.Type.NORMAL);
        when(itemRepository.findAll()).thenReturn(List.of(item));

        List<Item> itemsUpdated = qualityService.updateQuality();

        assertEquals( expected: 0, itemsUpdated.get(0).getId());
        assertEquals( expected: "Oreo", itemsUpdated.get(0).name);
        assertEquals( expected: 9, itemsUpdated.get(0).sellIn);
        assertEquals( expected: 29, itemsUpdated.get(0).quality);
        assertEquals(Item.Type.NORMAL, itemsUpdated.get(0).type);
        verify(itemRepository,times( wantedNumberOfInvocations: 1)).save(any());
    }

    @Test
    public void testUpdateQualityOfAgedTypeItem(){

        var item = new Item( id: 0,  name: "Oreo",  sellIn: 10,  quality: 30, Item.Type.AGED);
        when(itemRepository.findAll()).thenReturn(List.of(item));

        List<Item> itemsUpdated = qualityService.updateQuality();

        assertEquals( expected: 0, itemsUpdated.get(0).getId());
        assertEquals( expected: "Oreo", itemsUpdated.get(0).name);
        assertEquals( expected: 9, itemsUpdated.get(0).sellIn);
        assertEquals( expected: 31, itemsUpdated.get(0).quality);
        assertEquals(Item.Type.AGED, itemsUpdated.get(0).type);
        verify(itemRepository,times( wantedNumberOfInvocations: 1)).save(any());
    }
```

6. Finally, we updated the ItemController creating ab object associated to the class QualityService.



```java
@RestController
@RequestMapping(value = "api/items")
public class ItemController {

    private final ItemService itemService;
    private final QualityService qualityService;

    public ItemController(ItemService itemService, QualityService qualityService) {
        this.itemService = itemService;
        this.qualityService = qualityService;
    }

    @GetMapping()
    public ResponseEntity<List<Item>> listItems(){
        var items  :List<Item>  = itemService.listItems();
        return new ResponseEntity<>(items, HttpStatus.OK);
    }
```

```
[INFO] Installing C:\Users\Taborda\Perficent-Practice-1\pom.xml to C:\Users\Taborda\.m2\repository\com\perficient\pra
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  39.542 s
[INFO] Finished at: 2022-03-29T21:50:03-05:00
[INFO] ------------------------------------------------------------------------

Process finished with exit code 0
```