Projeto de DCO 2023/2024

Ricardo Fernandes n.º 59823
João Ferreira n.º 58191

Relatório feito por Ricardo Fernandes

April 19, 2024

Ricardo Fernandes n.º 59823
João Ferreira n.º 58191

Contents

Ricardo Fernandes n.º 59823
João Ferreira n.º 58191

1 Código

O output do projeto está bastante semelhante ao esperado. No entanto, existem algumas diferenças, algumas destas fora do nosso controlo:

1. No toString da classe MainLibrary,as fotos AnelJVasconcelos e Polvo **estão trocadas na ordem como aparecem**. Isto deve-se puramente ao facto que **o tamanho em bytes da foto AnelJVasconcelos é menor que o tamanho da foto Polvo, sendo, portanto, o output pretendido**. No entanto, no output que a professora colocou no moodle, a foto AnelJVasconcelos tem um maior tamanho que a foto Polvo. Esta situação não depende do nosso código, apenas resultando do tamanho lido pela classe responsável por obter os metadados das fotos. Assim, apesar de essa parte não estar igual ao output da professora, o nosso output está de acordo com esperado.

2. No toString da variável com o nome mostRecent presente na ViewsCatalog, a foto AnelJVasconcelos não aparece. Isto deve-se ao facto que **não foi possivel obter a data de captura desta foto**.**Assim, de acordo com o esperado, é atribuida a data 1970-01-01 à foto**. Logo, como para que a foto pertenca à library representada por esta variável a foto tem de ter sido capturada nos ultimos 12 meses, então a foto AnelJVasconcelos não vai ser adicionada, tal como esperado.

3. O toString de AAlbum não vai retornar as fotos presentes no album na ordem que aparece no output da professora. Isto deve-se ao facto que no output da professora, as fotos aparecem pela ordem que foram adicionadas ao album. No entanto, nós usámos um HashSet a fim de diminuir a complexidade temporal de varias operações.Como o HashSet nao promete iterar sobre os seus elementos por ordem de chegada,entao tambem não é garantido que o toString (e o getPhotos) de AAlbum retornem os elementos por ordem de chegada.

4. No output do nosso programa, existem algumas linhas em branco e espaços que não aparecem no output da professora. No entanto, como acreditamos que estas diferenças não afetam o programa de forma alguma,não perdemos tempo a alterá-las apenas para estar exatamente igual ao output da professora.

Além disto, eu fiz algo que acredito que mereça ser mencionado, devido ser algo que possa não ser intuitivo/natural à primeira vista, **esta sendo a decisão na complexidade temporal de vários métodos e operações da MainLibraryView**.

1.1 Decisão na alteração da complexidade temporal de vários métodos e operações na MainLibraryView

Alterações:

- processEvent: PhotoAddedLibraryEvent O(1) → O(log n)
- PhotoChangedLibraryEvent O(1) → O(1)
- PhotoDeletedLibraryEvent O(1) → O(log n)
- numberOfPhotos: O(1) → O(1)
- setComparator: O(1) → O(1)
- getPhotos: O(n log(n)) → O(n) (99.9% dos casos)
- O(n log(n)) → O(n log(n)) (0.1% dos casos)
- GetMatches: O(n log n) → O(n)

Esta decisão foi tomada devido ao facto de getPhotos e getMatches serem os métodos com uma probabilidade maior de serem utilizados. O aumento da complexidade temporal na adição de uma foto não vai ser muito drástico numa execução normal do programa, enquanto que a redução da complexidade temporal de getPhotos de O(n log(n)) para O(n) é bastante benefícial.

Por exemplo, se a view tiver 500 fotos entao, como a complexidade temporal de uma adição de foto é $\log_2(n)$, então seria $\log_2(500) = 9$. No entanto, anteriormente, a complexidade temporal de getPhotos era O(n $\log_2(n)$), o que significaria que para o mesmo numero de fotos(neste caso 500), este método demoraria 500 * $\log_2(500)$ = 500 * 9 = 4500. No entanto, como agora a complexidade temporal do getPhotos é O(n), o método demoraria 500!

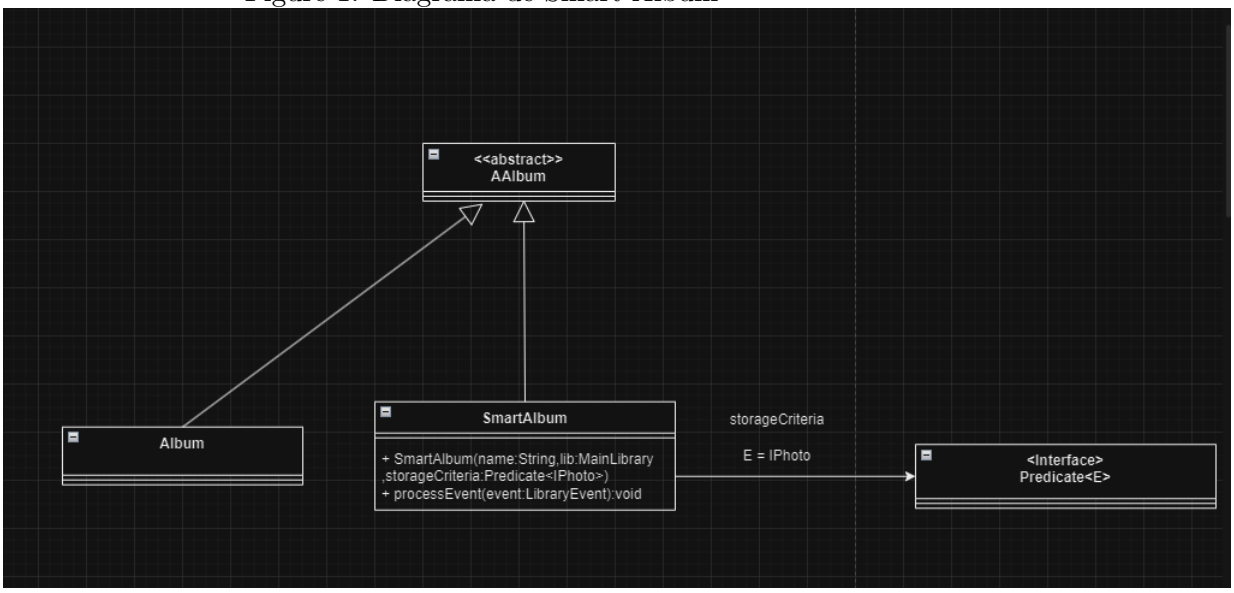
O pior dos casos seria se houvesse constantemente uma mudança de comparador. Como este caso é bastante improvável, decidi optar por esta implementação.

2 Desenho

2.1 Smart Album

2.1.1 Diagrama

Figure 1: Diagrama de Smart Album



2.1.2 Explicações/Esclarecimentos

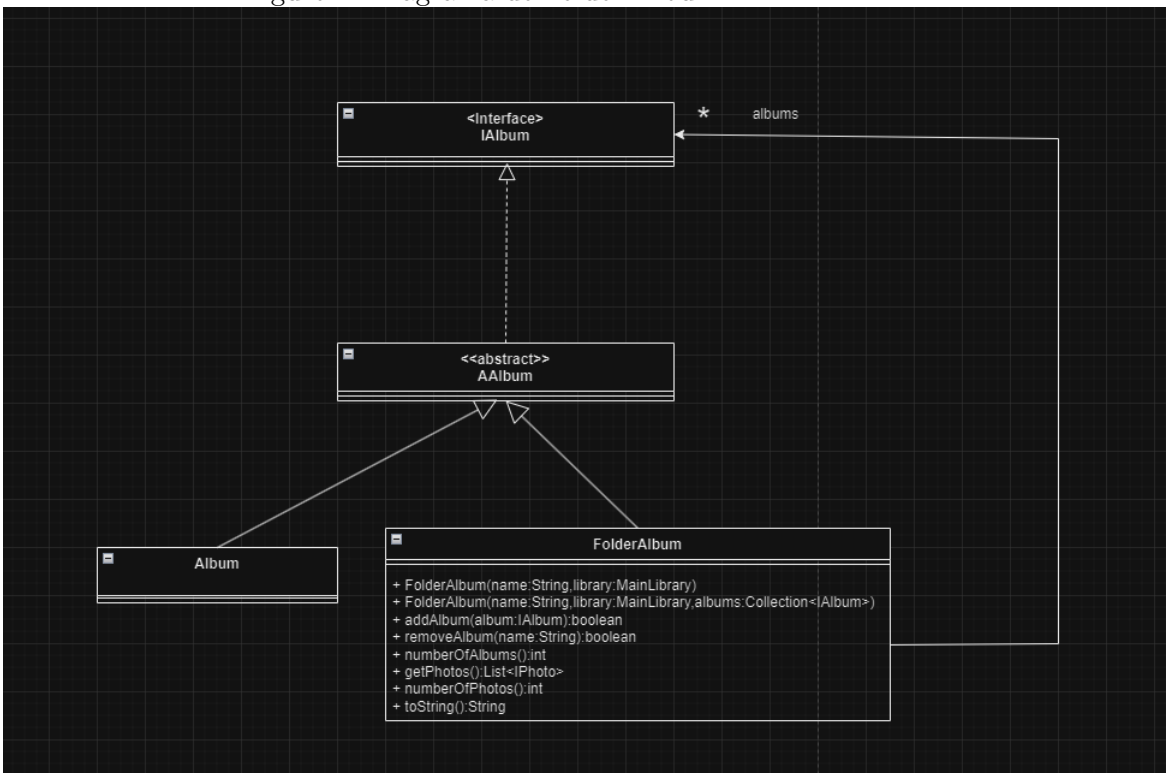
1. Esta classe vai "dar override" ao método processEvent presente em AAlbum pois, ao contrário do que acontece em AAlbum, **esta classe quer processar os 3 tipos de eventos** (PhotoAddedLibraryEvent,PhotoChangedLibraryEven e PhotoDeletedLibraryEvent)

Nota: Foram omitidos os métodos e atributos das classes AAlbum e Album a fim de poupar espaco no diagrama.

2.2 Folder Album

2.2.1 Diagrama

Figure 2: Diagrama de Folder Album



2.2.2 Explicações/Esclarecimentos

1. Aqui usou-se o **padrão composite** devido ao facto que FolderAlbum pode ter outros albums dentro dele e também funciona como um album. Ou seja, é um IAlbum que pode conter vários IAlbums.

2. Esta classe vai "dar override" ao método getPhotos da classe AAlbum.Isto deve-se ao facto que o getPhotos de AAlbum apenas retorna as fotos do album. No entanto,**como FolderAlbum pode conter outros albums, entao é necessario fazer com que também retorne as fotos dos outros albums**.Fizemos assim pois assumimos que este era o objetivo de getPhotos no FolderAlbum (devido ao que nos faz sentido e tambem devido a varios exemplos em que isto era o pretendido, alguns destes realizados nas aulas TP). Porém, se o getPhotos apenas devesse retornar as fotos do proprio album e nao dos albums que este contem, então este metodo poderia ser removido.

3. Esta classe tambem vai "dar override" ao método numberOfPhotos pela mesma razao que dá override a getPhotos (razão anteriormente mencionada). O numberOfPhotos de AAlbum apenas retorna o numero de fotos do proprio album e nao de albums que este contem. Assim, é necessário que esta classe defina uma nova implementação desse método, fazendo com que retorne o número de fotos do próprio album mais o número de fotos de todos os albums que este contem.

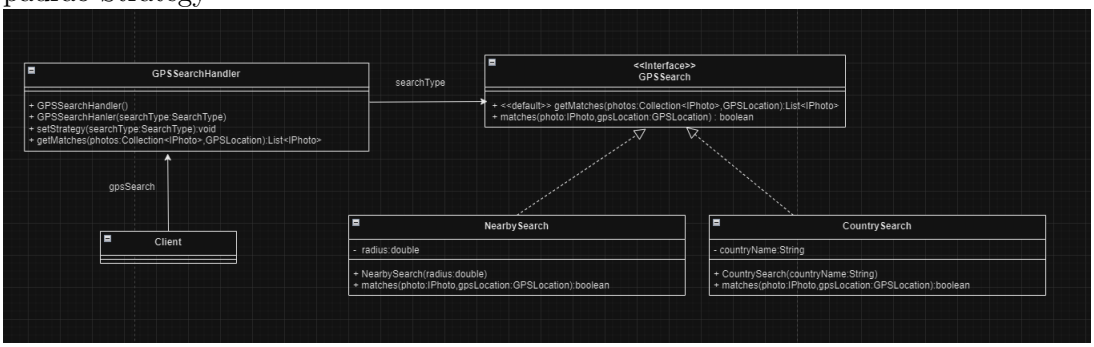
4. Esta classe também vai "dar override" ao método toString pela mesma razão que "da override" a getPhotos. O toString de AAlbum apenas retorna a representação textual do album contendo as fotos deste. No entanto, um toString apropriado de um FolderAlbum deveria também conter os albums que este contem e as fotos destes.
   Um exemplo de uma possivel representação textual seria:
   Folder Album: albumName
   foto1
   foto2
   foto3
   subAlbumName (aqui chamar-se-ia o toString do subAlbum)
   foto4
   foto5
   subAlbumName2 (aqui chamar-se-ia o toString do subAlbum)
   foto6
   End FolderAlbum: albumName

Nota: Foram omitidos os métodos e atributos das classes AAlbum, Album e os métodos da interface IAlbum,a fim de poupar espaco no diagrama.

2.3 Procura de fotos por localizacao gps - Padrao Strategy

2.3.1 Diagrama

Figure 3: Diagrama da procura de fotos por localização de gps usando o padrao Strategy



2.3.2 Explicações/Esclarecimentos

1. O padrão strategy **deve ser usado na eventualidade de apenas se poder fazer um tipo de procura por GpsLocation de cada vez**. No caso de se poder fazer mais em simultâneo, então deve-se usar a alternativa seguinte, presente no ponto 2.4.

2. **Usou-se o strategy pois o objetivo é ir alterando a variante de pesquisa por gps de acordo com o que queremos**.

3. A classe Client serve para representar qualquer classe que utilize a classe GPSSearchHandler.

4. A interface GPSSearch tem um método default, este sendo: matches(List¡IPhoto¿ photos,GPSLocation location).**Este método vai utilizar o padrão Template Method** pois, por cada elemento presente na coleção que recebe como argumento, vai chamar o método getMatches(IPhoto,GPSLocation) que é um método que terá de ser implementado pelas classes que implementem esta interface.Ou seja, o código deste metodo default seria algo assim:

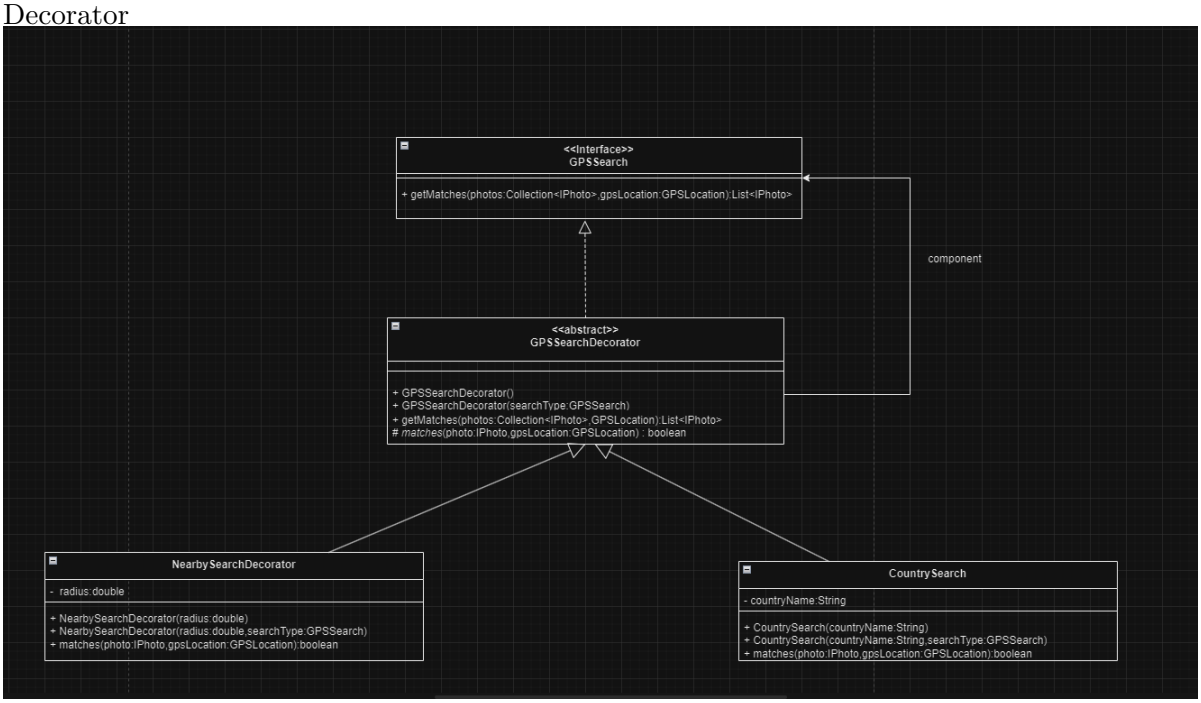
```
List<IPhoto> list = new ArrayList<>();
for(IPhoto photo : photos){
    if(matches(photo,gpsLocation)){
        list.add(photo);
    }
}
```

5. O construtor default de GPSSearchHandler poderá inicializar o objeto searchType como um objeto do tipo GPSSearch default.Se isto não for pretendido, pode-se apenas remover este construtor.

2.4 Procura de fotos por localizacao gps - Padrao Decorator

2.4.1 Diagrama

Figure 4: Diagrama da procura de fotos por localizacao gps usando o padrao Decorator



2.4.2 Explicações/Esclarecimentos

1. **O padrão decorator deve ser usado na eventualidade de se poder fazer vários tipo de procura por gps simultaneamente**. No caso de se apenas poder fazer um de cada vez , então deve-se usar a alternativa anterior, presente no ponto 2.3.

2. Usou-se o decorator pois o objetivo é juntar os matches dos vários tipos de pesquisa a fim de determinar se uma foto verifica estas várias condições.Isto pode ser útil por exemplo se um utilizador quiser **obter todas as fotos tiradas a 20km de uma certa coordenada de gps e num pais X**.

3. A classe abstrata GPSSearchDecorator utiliza o **padrão Template Method no método getMatches** que, por cada elemento da collection que recebe como argumento, vai chamar o método matches(IPhoto,GPSLocation) que é um metodo que terá de ser implementado pelas classes que herdam esta classe.