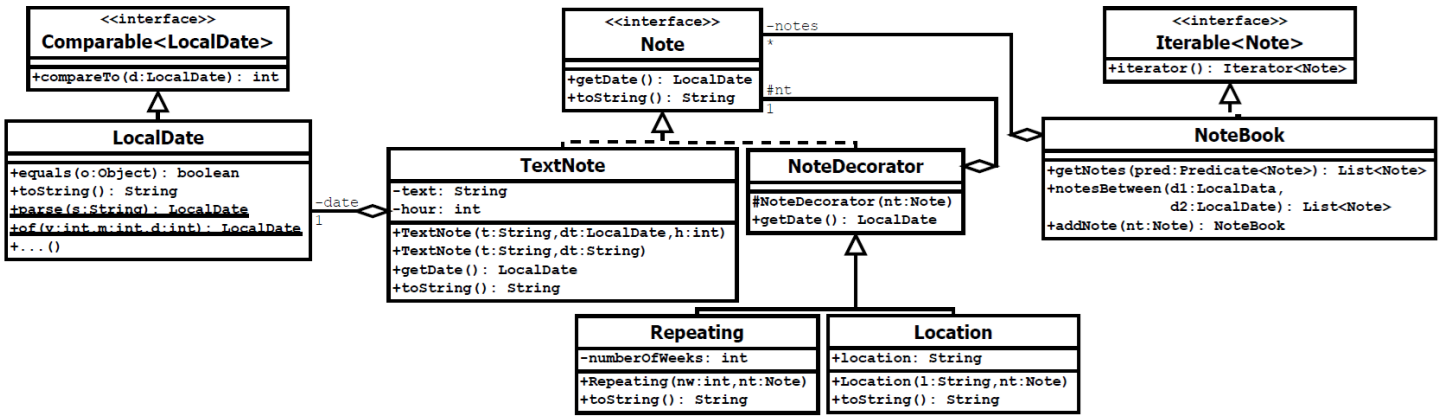
Instituto Superior de Engenharia de Lisboa
Programação III
1º Teste – 9 de Janeiro de 2024 (Duração: 1:00)

ISEL ADEETC

Para representar anotações foi definida a hierarquia apresentada na figura. A interface **Note** representa o contrato de qualquer tipo de anotação. As anotações mais simples têm apenas texto e uma data (classe **TextNote**). As anotações podem ter também um local (**Location**) ou serem repetitivas (**Repeating**). A classe abstrata **NoteDecorator** é a base para representar estas características, ou mais que possam existir no futuro. Cada característica que decora uma anotação é também uma anotação. A classe **NoteBook** agrega várias anotações.



Tendo em conta o diagrama estático de classes, os troços de código e os respetivos *outputs*, defina:

1. [4] A Interface **Note** e a classe **TextNote**. A classe **TextNote** no construtor recebe por parâmetro o texto, a data e a hora, ou só o texto e data. O seguinte troço de código exemplifica o que deve ser retornado pelo método **toString** tendo em conta os valores recebidos por parâmetro no construtor.

Nota: O método estático parse de **LocalDate** retorna a **LocalDate** que a *string* recebida por parâmetro representa.

```
Note nt1= new TextNote("Test PG3", LocalDate.of(2024,1,9), 9);
System.out.println( nt1 );
```

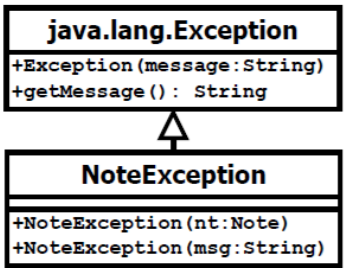
```
"Test PG3" 2024-01-09 09:00
```

```
Note nt2= new TextNote("Class PG3","2023-09-11");
System.out.println( nt2 );
```

```
"Class PG3" 2023-09-11
```

2. [2] A classe **NoteException** para que o método **getMessage** herdado de **Exception** retorne:

- A *string* "*Duplicate info:* " seguida da *string* retornada pelo método **toString** da anotação nt, caso tenha sido instanciado com o construtor com um parâmetro do tipo **Note**;
- A *string* passada por parâmetro no construtor caso tenha sido instanciado com o construtor com o parâmetro do tipo **String**.



3. [2] A classe abstrata **NoteDecorator**. No construtor recebe a anotação base. Tenha em conta que o campo **nt** está acessível nas classes derivadas, mas não no restante código. A data de um **NoteDecorator** é a data da anotação que é decorada (**nt**).

4. [2] A classe **Repeating**. No construtor recebe o número de semanas e a anotação base, lança a exceção **NoteException** se o número de semanas for menor que 2 passando a mensagem *"Invalid number of weeks*". O método **toString** retorna a *string* da anotação **nt**, referida no construtor, concatenada com a *string ", repeating for N weeks*" sendo N o valor passado no construtor.

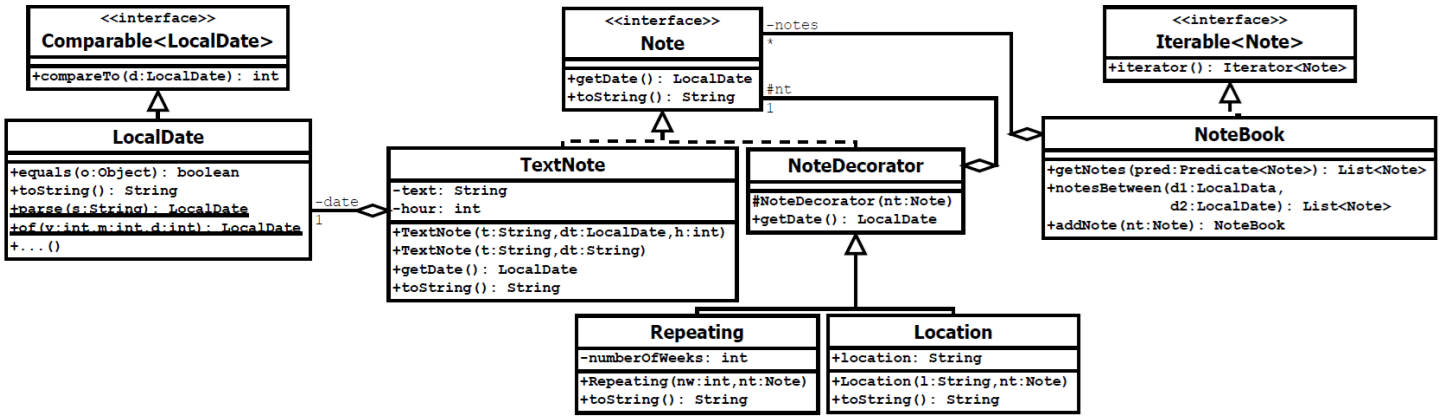
```
Note nt3 = new Repeating(15, new TextNote("Class PG3", "2023-09-11"));
System.out.println(nt3);
```

```
"Class PG3" 2023-09-11, repeating for 15 weeks
```

5. [2] A classe **Location.** No construtor recebe o local e a anotação base. O método **toString** retorna a *string* da anotação **nt**, referida no construtor, mais a informação "@ L" sendo L a *string* passada no construtor. O campo **location** embora público não pode ser alterado após a instanciação.

```
Note nt4 = new Location("E1.12", new TextNote("Class PG3", "2023-09-11"));
System.out.println(nt4);
```

```
"Class PG3" 2023-09-11, @ E1.12
```



6. [6] A classe **NoteBook** tendo em conta que:

- O método **getNotes** retorna uma lista, ordenada por datas, com as anotações que obedecem ao predicado.
- O método **notesBetween** retorna uma lista com as anotações cuja data é maior do que **d1** e menor do que **d2**. Na implementação do método chame o método **getNotes** e implemente um **Predicate<Note>**.
- O método **addNote** caso já exista uma anotação com a mesma data lança a exceção **NoteException** passando-lhe por parâmetro a anotação. Caso contrário adiciona-a e retorna a própria agenda. Para verificar se já existe implemente um **Predicate<Note>** e chame o método **getNotes**.
- O método **iterator** retorna o **Iterator<Note>** para as anotações da agenda.

```
NoteBook nb = new NoteBook();
try{
  Note nt1= new TextNote("Test PG3",LocalDate.of(2024,1,9), 19);
  Note nt2 = new Location("E1.12",
              new Repeating(15,
                new TextNote("Class PG3", "2023-09-11")));

  nb.addNote( nt1 ).addNote( nt2 );

  nb.addNote( nt1 );
} catch(NoteException ex) {
   System.out.println( ex.getMessage() );
}
```

```
Duplicate info:  "Test PG3" 2024-01-09 19:00
```

```
// Lista pela ordem de adição
for( Note nt: nb )
  System.out.println( " -> " + nt );
```

```
-> "Test PG3" 2024-01-09 19:00
-> "Class PG3" 2023-09-11, repeating for 15 weeks, @ E1.12
```

```
// Listagem ordenada por datas
List<Note> notes = nb.getNotes( n -> true );
for( Note nt: notes )
  System.out.println( " -> " + nt );
```

```
-> "Class PG3" 2023-09-11, repeating for 15 weeks, @ E1.12
-> "Test PG3" 2024-01-09 19:00
```

7. [2] Usando as anotações das alíneas anteriores, instancie uma anotação onde o método **toString** retorne a seguinte *string*:

```
"Playday" 2024-08-01, @ beach, countryside, …, repeating for 4 weeks
```