Exercise *printer for time tracker*

1 The problem

In the TimeTracker project we have the task and project classes, derived from the activity class. Also, a task may have intervals. At some point in the development we want to check that times and durations are correct. The easiest way to do it is to add some print sentences at the method where a project, task and interval is updated, from bottom (interval) to top (root project). Hence, we'll get prints only for the objects currently being updated.

A better way if the tree is not too large is to periodically print the whole tree in a certain order like top-down, say every 2 seconds. By printing the whole tree we can also make sure objects that must not change, don't.

There is a trick to quickly check that the result is the one desired: since you are sending the output to the console (i.e. `System.out.println`), adjust the height of the console window so that it shows the same number of lines as there are objects in the tree. It will seem there is no scrolling and you can quickly spot and read the things that change.

We want to print the task/project name, initial and final date and duration, and same for intervals except of course for the name. Also, we want to print the class of the object:

```
Project root    child of null  31-10-2021 14:35:18  31-10-2021 14:35:48  284
Project P1      child of root  31-10-2021 14:35:18  31-10-2021 14:35:48  284
Task T1         child of P1    31-10-2021 14:35:18  31-10-2021 14:35:48  284
Interval        child of T1    31-10-2021 14:35:18  31-10-2021 14:35:28  160
Interval        child of T1    31-10-2021 14:35:40  31-10-2021 14:35:48  124
Project P2      child of root  null                 null                 0
```

In the future we will need to add several other processes that, like this one of printing, need to traverse the tree and do different things depending on the type of the process *and* the type of object (task, project, interval). We look for a solution to the printing problem that can also accommodate the new processes in order to keep a high cohesion, that is, following the single responsibility principle.

An elegant way to implement it is by combining two design patterns. One is in charge of the periodicity of printing, and doing it simultaneously to the user interaction. This means the printing command is sent from a different thread. The other pattern avoids to include the code of how to print the particular object in its class, which is the natural way to do it according to the expert pattern (who has the information to do something is the responsible of doing it) and because this depends on its class.

2 What to do

Design the solution to the periodic printing requirement. Make a detailed class diagram in PlantUML. Then program it in a new IntelliJ project independent of the practicum. Write testing code to show it works. Include in the IntelliJ project the PlantUML diagram and a text file containing the output to console of the test.

3 FAQ

Q: El diagrama UML s'ha de fer de tota la pràctica o només sobre la part que afecta el "printer"?

A: Al disseny ha de sortir, de la pràctica, el rellotge, tasques, projectes, Observer i Observable. I a més les coses que afegeixis per aquest exercici: una classe Printer que hereta o implementa Visitor.

Q: Com d'avançada ha d'estar la practica per a poder fer aquest exercici?

A: Necessiteu que funcioni el més bàsic de la fita 1: poder crear un arbre de tasques i projectes (Composite) i poder comptar el temps.

Q: Fem servir el mateix projecte IntelliJ de la pràctica per a realitzar aquest exercici?

A: El que heu de fer és copiar a un altre projecte aquesta part de la pràctica, i llavors afegir-hi la part d'impressió. Si encara no teniu feta la part de comptar el temps (incloent-hi interval, tasca i projectes antecessors), podeu fer l'impressor també, que sempre imprimirà el mateix, i després afegir-li aquesta part per acabar-ho.

Q: Com seria un possible "main" per a demostrar el funcionament del "printer"?

A: De forma similar a com feu a la pràctica:

```java
public class Main {
  public static void main(String[] args) throws
   InterruptedException {
    // create and start the clock
    //...

    // make a small tree of projects and tasks
    Project root = new Project("root", null);
    Project p1 = new Project("P1", root);
    Project p2 = new Project("P2", root);
    Task t1 = new Task("T1", root);
    Task t2 = new Task("T2", p1);
    Task t3 = new Task("T3", p2);

    // make the printer
    Printer printer = new Printer(root);

    // the printer will periodically print the whole tree
    // from now on
    // ...

    // test it
    Thread.sleep(4000);
    // this will make some intervals
    t1.start();
    Thread.sleep(4000);
    t2.start();
    Thread.sleep(2000);

    // optionally, stop the clock
    // ...
  }
}
```

En executar aquest codi s'haurà de mostrar al stdout una sortida (periòdica) de l'estat de l'arbre, similar a aquesta:

2021-10-17T19:32:18.130664				
Project root	child of null	null	null	0
Project P1	child of root	null	null	0
Task T2	child of P1	null	null	0
Project P2	child of root	null	null	0
Task T3	child of P2	null	null	0
Task T1	child of root	null	null	0
2021-10-17T19:32:20.224866				
Project root	child of null	2021-10-17T19:32:18.224866	2021-10-17T19:32:20.224866	2
Project P1	child of root	null	null	0
Task T2	child of P1	null	null	0
Project P2	child of root	null	null	0
Task T3	child of P2	null	null	0
Task T1	child of root	2021-10-17T19:32:18.224866	2021-10-17T19:32:20.224866	2
Interval	child of task T1	2021-10-17T19:32:18.224866	2021-10-17T19:32:20.224866	2
2021-10-17T19:32:22.251584				
Project root	child of null	2021-10-17T19:32:18.224866	2021-10-17T19:32:22.251584	4
Project P1	child of root	null	null	0
Task T2	child of P1	null	null	0
Project P2	child of root	null	null	0
Task T3	child of P2	null	null	0
Task T1	child of root	2021-10-17T19:32:18.224866	2021-10-17T19:32:22.251584	4
Interval	child of task T1	2021-10-17T19:32:18.224866	2021-10-17T19:32:22.251584	4
2021-10-17T19:32:24.253682				
Project root	child of null	2021-10-17T19:32:18.224866	2021-10-17T19:32:24.253682	8
Project P1	child of root	2021-10-17T19:32:22.253682	2021-10-17T19:32:24.253682	2
Task T2	child of P1	2021-10-17T19:32:22.253682	2021-10-17T19:32:24.253682	2
Interval	child of task T2	2021-10-17T19:32:22.253682	2021-10-17T19:32:24.253682	2
Project P2	child of root	null	null	0
Task T3	child of P2	null	null	0
Task T1	child of root	2021-10-17T19:32:18.224866	2021-10-17T19:32:24.253682	6
Interval	child of task T1	2021-10-17T19:32:18.224866	2021-10-17T19:32:24.253682	6

Table 1:

4 Deliverables

As usual: authors' file, source code, plantUML class diagram (source and image), console output, a brief text explaining the solution.

5 Grading

You can get 0, 0.25 or 0.5 points.