Opgave 1 : Threadklasse en Tekstbestand

Schrijf een Java-applicatie waarin je

1) een **Threadklasse** toepast (zie Laan, H13)
2) Een **tekstbestand** leert schrijven (zie Laan, H14)

1. Begin met de Thread:

Maak een class **Klokje**, die onafhankelijk van je hoofdprogramma continu de tijd weergeeft. Waar dit gebeurt mag je zelf kiezen : in een apart JFrame, in een JOptionPane, of in een van de vijf JPanels van een BorderLayout op je hoofdpaneel.
(zoek de nodige code via de Java Index > dan de juiste klasse, en kijk daar of er een Tutorial voor bestaat)

TIP 1 : als je deze Thread klasse als interne klasse maakt in je hoofdprogramma, kan deze aan alle attributen (vb. de panelen) van je hoofdprogramma. Via de methode getGraphics() kan je dan de graphics context gebruiken om te tekenen. Enkel het wissen van het paneel moet je dan doen door een gevulde rechthoek in de achtergrond kleur te tekenen.
Maak je de Thread klasse als een externe klasse (zo hoort het eigenlijk) , dan moet je bij het instantiëren van de "Klok"-thread de identifier van het klokpaneel meegegeven, zodat je via de opdracht *getGraphics()* aan de grafische context geraakt. Maak in dat geval in je Thread klasse een geschikte Constructor.

TIP 2 : de tijd haal je gemakkelijkst uit een **Calendar** object, dit maak je als volgt :

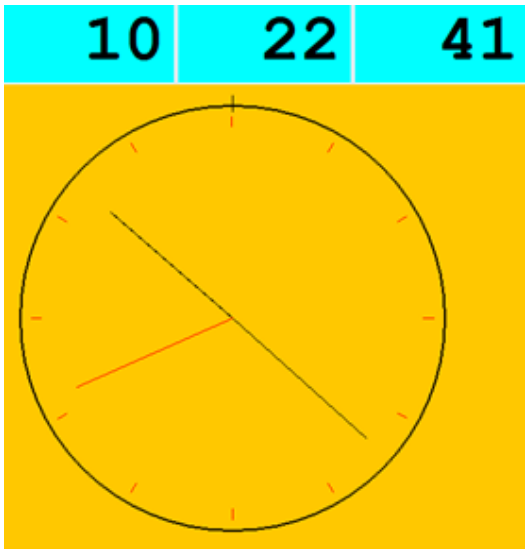
```
Calendar cd;
cd = Calendar.getInstance();  //*
```

de uren, minuten en seconden verkrijg je dan bvb. met :

```
int uren = cd.get(Calendar.HOUR_OF_DAY); // voor een 24-uurs klok
```

en idem met MINUTE en SECOND (en desgewenst MILLISECOND bij chronometertoepassingen)
Let op : om de tijd telkens actueel te houden moet je * steeds opnieuw uitvoeren (zie index).

Voorbeeld van een oplossing met digitale & analoge klok →
Zowel de digitale als de analoge klok worden dus vanuit dezelfde thread aangepast (om de seconde).



2. Database met bestandsopslag.
Als de klok werkt maak je een mini-database programma, met voorlopig opslag in een tekstbestand.

Baseer je losjes op het CD voorbeeld uit H7.
Schrijf nog een aparte klasse die je “data-object” beschrijft (CD, boek, persoon, of wat je ook maar wil bewaren).
Voorzie enkele tekstvakken om de data in te voeren + een knop om het object in een ArrayList toe te voegen.
Voorzie ook een knop waarmee alle data uit de ArrayList naar een tekstbestand weggeschreven wordt, 1 lijn per object. TIP : voorzie je data-klasse van een toString() methode !
Controleer via kladblok of je data correct weggeschreven werd.
De naam voor het bestand kan je voorlopig via een tekstvak invoeren, dit laat de mogelijkheid om een verkeerd path in te voeren en een FOUT te forceren (zie 3. EXTRA).
(vb F:/temp/data.txt, terwijl je geen F-schijf hebt).

3. EXTRA : in je programma heb je allicht één of meerdere try-catch blokken , voorzie dat je in minstens één catch-blok de fout die je opgevangen hebt in een zogenaamd “error.log” bestand bijschrijft. Gebruik hierbij de append modus (tweede argument in de constructor = true), zodat je de vorige foutmelding laat staan.
Zet bij elke fout ook datum en tijd.

Dien voor het einde van de oefenzitting een verslag in via e-mail zoals in de **algemene richtlijnen** beschreven staat, ook al is je programma nog niet helemaal af.
Zet stukken die eventueel nog niet helemaal werken in commentaar, en vermeld dit in je verslag.

Veel succes !