Examen MOCI 2014

TELECOM Nancy - 2A - 2014/2015

Durée 2h

Documents autorisés : Une feuille A4 recto-verso d'aide-mémoire et le recto/verso sur les patrons de conception

La qualité des diagrammes UML et le respect de la notation seront pris en compte dans l'évaluation de chaque question où UML est utilisé.

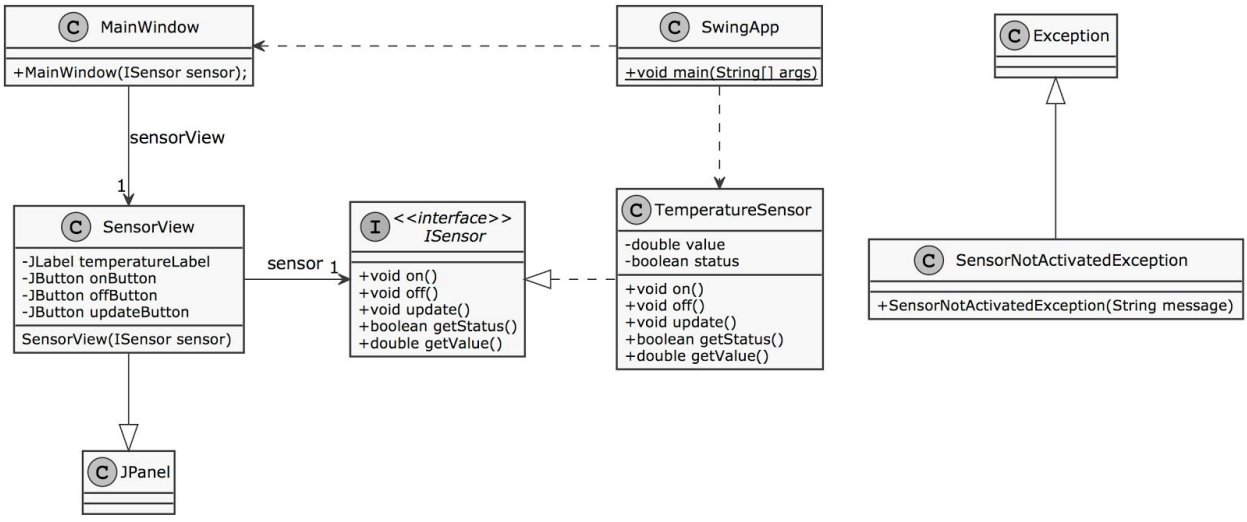
1) Ingénierie des besoins (5 points)

I - Un client veut mettre en place un système de vente de tickets de trains. Le prix d'un ticket est calculé selon la distance à parcourir. Les utilisateurs de ce système peuvent choisir une destination et payer avec une carte de crédit. Le paiement est assuré par un organisme bancaire tiers.

a. écrivez deux besoins fonctionnels pour ce système.
b. écrivez deux besoins non fonctionnels en indiquant comment vous pourrez vérifier qu'ils sont validés.

II - Que doit-on vérifier lors de la phase de validation des besoins ?

2) UML (5 points)



I - Sur la base de l'application vue en TP, faites un diagramme de séquence correspondant au démarrage de l'application SwingApp en vous assurant de bien respecter la notation UML.

II - Que permettent de modéliser les diagrammes d'activité UML ? Donnez un exemple.

3) Conception (6 points)

I - Nommez deux types de couplage entre deux classes A et B.

II - A quoi correspond le pattern Creator de GRASP ? Donnez un exemple où il est recommandé de le mettre en oeuvre dans le cas de l'application ci-dessus.

III - Dans l'application que vous avez développée en TP, on veut utiliser le patron Commande pour contrôler les capteurs.

a. rappelez l'intérêt de ce patron
b. faites le diagramme de classe en étendant le diagramme de classe ci-dessus
c. faites le diagramme de séquence de l'allumage de capteur (commande On) avec ce patron

4) Architecture (4 points)

I - Qu'est-ce-qu'un style d'architecture dans le domaine du logiciel ? Donnez un exemple autre que MVC.

II - Faites le diagramme d'une architecture MVC en indiquant bien le rôle et les relations entre les différents composants.

```
@startuml
interface ISensor <<interface>> {
  +void on()
  +void off()
  +void update()
  +boolean getStatus()
  +double getValue()
}

class TemperatureSensor implements ISensor {
  - double value
  - boolean status

  + void on()
  + void off()
  + void update()
  + boolean getStatus()
  + double getValue()
}

class SensorView extends JPanel {
  - JLabel temperatureLabel
  - JButton onButton
  - JButton offButton
  - JButton updateButton

  SensorView(ISensor sensor)
}

SensorView -> "1" ISensor : sensor

class MainWindow {
  + MainWindow(ISensor sensor);
}

MainWindow --> "1" SensorView : sensorView

class SwingApp {
  +{static} void main(String[] args)
}

SwingApp ..> MainWindow
SwingApp ..> TemperatureSensor

@enduml
```