420-V41-SF
PROGRAMMATION D'APPLICATIONS MOBILES I
ÉVALUATION FINALE À CARACTÈRE SYNTHÈSE – PARTIE 3
L'ACTIVITÉ DE TRAITEMENT DES DONNÉES D'HUMIDITÉ
PONDÉRATION : 13%

OBJECTIFS PÉDAGOGIQUES

Ce travail vise à familiariser l'étudiante ou l'étudiant avec les objectifs suivants :

- 016T- Appliquer une approche de développement par objets.
- 017D- Concevoir et développer une application hypermédia dans des réseaux internes et mondiaux.
- 0177- Assurer la qualité d'une application

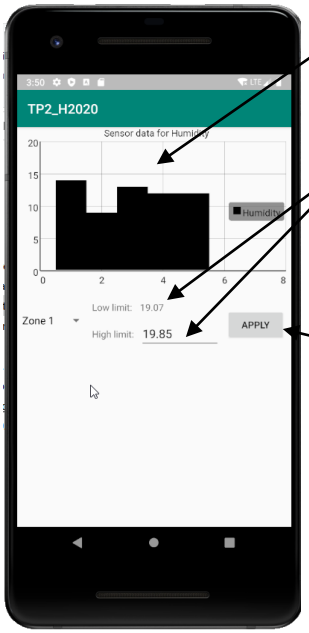
De même, il permet à l'étudiante ou à l'étudiant d'appliquer les standards de conception, de documentation et de programmation.

MISE EN CONTEXTE ET MANDAT

Pour la troisième partie de ce dernier travail, vous devez réaliser une activité Android qui permet de traiter les données d'humidité acquises dans la première partie de ce travail. Veuillez noter qu'il vous est encore possible de simuler les données dans l'éventualité où votre première partie n'était pas totalement fonctionnelle.

CONTRAINTES DE RÉALISATION

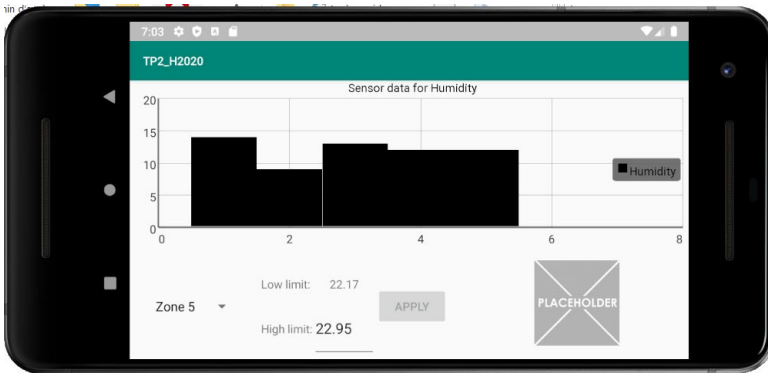
Pour la partie 3 du travail, vous devez produire une application Android offrant les fonctionnalités suivantes :



L'application doit afficher un graphique des niveaux d'humidité observés par regroupement. Pour cela, utilisez la librairie `com.jjoe64:graphview:4.2.2`.

- L'application doit permettre de spécifier des limites hautes (et basses) pour chaque regroupement. Par défaut, la limite haute de la catégorie précédente constitue la limite basse de la catégorie suivante. La limite haute de la dernière catégorie est la plus haute valeur trouvée dans les données reçues.
- Il doit être possible de spécifier dynamiquement la limite haute à l'aide du bouton approprié. Les modifications sont apportées à la limite concernée et les regroupements suivants sont « divisés » selon la règle suivante : étendue des données restantes / nombre de regroupements restants.

- En aucun cas, la limite inférieure ne peut être plus grande que la limite supérieure. Advenant cette possibilité, le bouton « appliquer » devra être désactivé.
- En aucun cas, la limite supérieure d'une zone ne peut être plus grande que la limite supérieure de la zone suivante. Advenant cette possibilité, le bouton « appliquer » devra être désactivé.
- Il n'est pas permis de modifier la limite supérieure de la zone 5. Ce sera toujours par défaut la valeur maximale de l'ensemble des données. Toute tentative visant à modifier cette limite entraînera le lancement de l'exception `InvalidHighLimitException`.
- L'activité d'humidité doit supporter les changements d'orientation :



- L'activité doit supporter les changements de langue. Cela dit, pour les besoins de ce travail, ne testez votre application qu'en anglais.
- Votre code doit être organisé avec le patron de conception MVC.
- Vous devez réaliser les tests unitaires du contrôleur et du modèle. A cette fin, vous devrez obligatoirement utiliser les méthodes suivantes :
    - Modèle :
        - `Zone getSelectedZone();`

          Cette méthode est appelée pour obtenir la zone présentement sélectionnée.
        - `void setSelectedZone(Zone selectedZone);`

          Cette méthode est appelée pour modifier la zone présentement sélectionnée.
        - `double getUpperLimit (Zone zone);`

          Cette méthode est appelée pour obtenir la limite supérieure de la zone spécifiée.
        - `double getLowerLimit (Zone zone);`

          Cette méthode est appelée pour obtenir la limite inférieure de la zone spécifiée.
        - `int getCountInZone (Zone zone);`

`Zone` est une **énumération** pouvant prendre les valeurs `ZONE_1` à `ZONE_5`.

Cette méthode est appelée pour calculer le nombre de mesures dans la zone spécifiée.

- `void setCurrentZoneHighLimit(double newLimit);`

  Cette méthode est appelée pour modifier la limite supérieure de la zone présentement sélectionnée.

- `double getHighestValue();`

  Cette méthode doit retourner la valeur maximale de l'ensemble des données.

- Contrôleur :
  - `onZoneSelected(Zone zone);`

    Cette méthode serait appelée lors de la sélection d'une zone dans l'interface utilisateur. Elle est responsable de notifier le modèle du changement de zone et de mettre à jour les informations appropriées de la vue.

  - `onCurrentLimitChanged(double newLimit);`

    Cette méthode est appelée lorsque le contenu du Textbox de la limite supérieure est modifié. Son rôle est d'ajuster l'accessibilité du bouton « appliquer » dans l'interface utilisateur.

- Interface `HumidityView` :
  - `void update();`

    Cette méthode est appelée pour mettre à jour les informations affichées dans la vue.

  - `void setApplyButtonEnabled(boolean enabled);`

    Cette méthode est appelée pour ajuster l'accessibilité du bouton « appliquer » dans la vue.

  - `boolean isApplyButtonEnabled();`

    Cette méthode est appelée pour obtenir l'accessibilité du bouton « appliquer » dans la vue.

Vous serez évalués(ées) de la manière suivante : 1) Complétion de la suite de tests et 2) Réalisation d'une suite de tests que nous fournirons (d'où l'importance de respecter la signature des méthodes spécifiées juste avant).

- Vous devez commenter le code en mode javadoc. La forme est ce qui est le plus important, mais le fond doit décrire avec justesse ce que vous faites et ce, **en français**. Vous devrez générer et remettre votre javadoc.

L'opérationnalité des fonctionnalités ne garantit pas la cote de passage. L'architecture et la qualité du code réalisé ainsi que des documents complet et professionnel sont primordiaux. Si le professeur le juge à propos, toute étudiante ou tout étudiant pourra être convoqué à

une rencontre d'évaluation sur Teams pour vérifier son degré d'acquisition des connaissances et d'appréhension de la solution proposée.

CONTEXTE DE RÉALISATION ET DÉMARCHE DE DÉVELOPPEMENT

Ce travail pratique doit être réalisé **seul**.
Les pages qui suivent résument les étapes du projet ainsi que les biens livrables devant être remis à la fin de chacune d'elles.

Étape 1 : Analyse de la situation

Dans cette étape, vous devez prendre connaissance du mandat qui vous est confié.

Biens livrables :

- Aucun.

Méthode:

- s/o

Étape 2 : Programmation de l'application

Vous devez procéder à la codification des fonctionnalités demandées ainsi que des tests spécifiés.

Biens livrables :

- Code source de l'application documenté (javadoc). Vous devez aussi générer la javadoc dans un dossier à part.

Méthode:

- Vous devez remettre votre code source dans une archive .zip identifiée en fonction de votre nom. Par exemple, pour l'étudiant nommé Pierre Poulin, l'archive devrait être nommée `EFCS3_PierrePoulin.zip`.
- Vous devez aussi remettre le fichier `RemiseEFCS – Partie 3.docx` accompagné de la grille d'auto-évaluation correctement complétée.

  Cette archive doit être déposée sur LÉA **avant le 11 mai 2020 à 7h59**.

MODALITÉS D'ÉVALUATION

Vous trouverez dans le fichier `RemiseEFCS – Partie 3.docx` la grille de correction qui sera utilisée pour le travail. **Cette grille indique la pondération accordée à chacune des parties du projet**.

- Tous les **biens livrables** de chacune des étapes devront être **remis à temps** et selon les modalités spécifiées.