**E3-FI intervenant : Denis MADEC**

Contrôle de connaissances sur IHM

Devoir individuel sur machines, tout document autorisé. Communications interdites entre étudiants, tous moyens confondus.   
Le projet Eclipse complet devra être rendu sur **BlackBoard** ET à l’adresse [denis.madec@orange.fr](mailto:denis.madec@orange.fr) dans **un seul fichier zip** à votre nom-prenom au format ***NOM\_prenom.zip*** avant la fin de séance  
Le nom de certaines classes devra être suffixé par vos initiales (remplacer XX par vos initiales)  
Le non-respect de cette consigne entraînera des pénalités

projet Minuteur

Un projet nommé Minuteur est fourni**, qu’il faudra importer dans Eclipse puis renommer en Minuteur\_*NOM\_prénom***

Ce projet doit permettre de saisir une durée en secondes, puis de démarrer le décomptage par un clic sur le bouton Départ. Lorsque le décomptage arrive à 0, une sonnerie retentit.

Attention ! L’exécution du projet nécessitera l’option suivante en argument de la VM :

**--add-modules=javafx.controls,javafx.fxml,javafx.media**

La couche modèle du projet est déjà fournie, à laquelle il conviendra de ne pas effectuer de modifications.

Il restera à compléter les classes **com.dma.ModeleXX**, **com.dma.controleur.MinuteurControleurXX** et à créer la vue **com.dma.vue.MinuteurXX.fxml** avec l'outil SceneBuilder.

1. Conception de la vue **MinuteurXX.fxml**

Le conteneur racine doit être de type **BorderPane**

Le conteneur racine doit être de type **BorderPane** dimensionné à 600 (largeur) x 500 (hauteur)

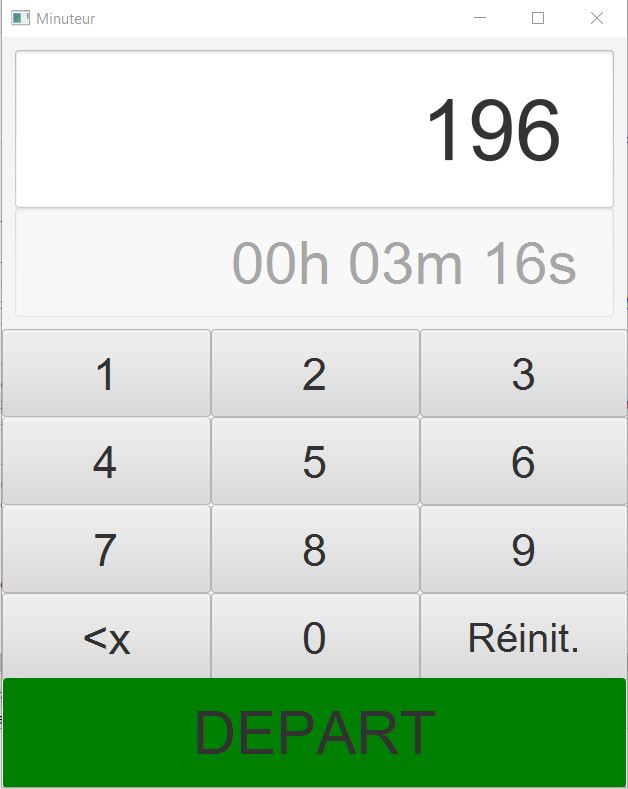
glisser-déposer un **TextField** en haut du **BorderPane**, avec alignement à droite,dimensionné à 480 (largeur) x 140 (hauteur). Sa police de caractères sera Arial, de taille 68 pt

glisser-déposer un **TextField** sous le premier **TextField,** dimensionné à 480 (largeur) x 50 (hauteur). Sa police de caractères sera Arial, de taille 48 pt

glisser-déposer sous le deuxième **TextField** un **GridPane** comportant 4 lignes de 3 colonnes, dimensionné de façon à occuper toute la largeur disponible. Y ajouter ensuite les 12 boutons, représentant les chiffres de 0 à 9 et le backspace, ainsi que Réinit. Tous ces boutons auront une police de caractères Arial, de taille 36 pt

glisser-déposer sous le **GridPane** un **Button** prenant toute la largeur de la fenêtre, de libellé **DEPART**, depolice de caractères Arial, de taille 48 pt. Sa couleur sera initialement verte

D'autres conteneurs (Hbox, Vbox, …) pourront être utilisés pour faciliter la mise en page



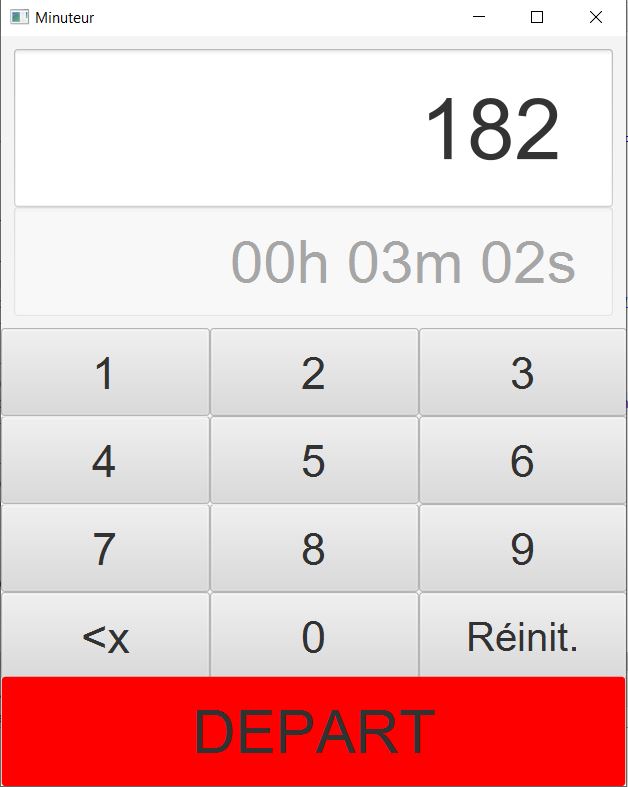
1. Conception du contrôleur **com.dma.controleur.MinuteurControleurXX**

Le contrôleur **MinuteurControleurXX** associé à la vue, devra permettre la saisie, dans le champ de texte supérieur, de 9 chiffres au maximum représentant une durée en secondes.

Le champ de texte inférieur sera invalidé en saisie, et devra représenter à tout instant en heures, minutes, secondes la durée apparaissant dans le champ de texte supérieur.

Le bouton DEPART permettra de démarrer le décomptage par appel à une méthode appropriée du modèle. Lorsque le décomptage est commencé, le champ de texte supérieur affiche la durée restante, le champ de texte inférieur la même durée en heures, minutes, seondes, le bouton DEPART devient rouge, et toute saisie dans le champ de texte supérieur est interdite.

Lorsque le décomptage arrive à son terme, une sonnerie retentit. Il faut actionner le bouton **Réinit** pour autoriser la saisie d'une nouvelle valeur.



1. Conception du modèle **com.dma.modele.MinuteurModeleXX**

La classe modèle **MinuteurModeleXX** devra être complétée pour la mise en œuvre du binding permettant le calcul de la durée en heures, minutes, secondes

package com.dma.modele;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.Timer;

import javafx.beans.property.BooleanProperty;

import javafx.beans.property.IntegerProperty;

import javafx.beans.property.SimpleBooleanProperty;

import javafx.beans.property.SimpleIntegerProperty;

import javafx.scene.media.AudioClip;

public class Modele implements ActionListener {

private Timer timer;

private IntegerProperty duree;

private BooleanProperty fin;

private AudioClip sonnerie;

public Modele() {

timer = new Timer(1000, this);

duree = new SimpleIntegerProperty(0);

fin = new SimpleBooleanProperty(false);

sonnerie = new AudioClip(this.getClass().getResource(

"/resources/sons/simple-office-tone.mp3").toExternalForm());

}

public IntegerProperty dureeProperty() {

return duree;

}

public BooleanProperty finProperty() {

return fin;

}

public void reinit() {

duree.setValue(0);

fin.setValue(false);

if(timer !=null && timer.isRunning()) {

timer.stop();

}

if(sonnerie.isPlaying()) {

sonnerie.stop();

}

}

public void depart() {

if(!timer.isRunning()) {

timer.start();

fin.setValue(true);

}

}

@Override

public void actionPerformed(ActionEvent arg0) {

if(duree.getValue() > 0) {

duree.setValue(duree.getValue() - 1);

}else {

timer.stop();

sonnerie();

}

System.out.println("duree restante: "+duree.getValue());

}

private void sonnerie() {

if(sonnerie != null) {

System.out.println("sonnerie....");

sonnerie.play();

System.out.println("fin de sonnerie....");

}

}

}