|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | **Informaticien/-ne CFC**  Travail pratique individuel 2024 (TPI) |  | |
|  | |

Modèle de rapport v1.4

Candidat No

Nom du candidat : Elvin Kuci

Attention : Dernier délai de remise du rapport durant la réalisation du TPI le 18.06.2021 à 17h00 hormis la présentation (ce texte doit être effacé).

**« Titre »**

### Sommaire

1 Travaux préparatoires 2

1.1 Installation de l’environnement 2

1.2 Création d'un projet JavaFX 2

1.3 JavaFX-TT 3

1.4 Moving-TT 5

Le Résumé du rapport du TPI est une présentation conceptuelle du travail effectué et du résultat attendu d’au maximum une page A4 qui permettent au lecteur une rapide compréhension du rapport de travail.. Il contient trois paragraphes : Situation de départ, mise en œuvre, résultats.

Pour plus de détails, veuillez vous référez au Manuel ICT - partie B : Documentation / rapport du TPI - Question 14 - page 81.

# Travaux préparatoires

## Installation de l’environnement

### Git

Pour installer Git, allez sur le site web [git-scm](https://git-scm.com/download/win) et installer la version voulue.

TODO : Mise en place du compte GitHub

### MySQL

TODO : Installation de MySQL ?

### Préparation et test de l’hébergement

TODO : Test et prep de l’hébergement

### Mise en place d’un projet JavaFX

Pour ce projet, NetBeans va être utilisé comme IDE. Vous pouvez installer NetBeans depuis le [site web officiel](https://netbeans.apache.org/front/main/download/).

Ensuite, pour créer un projet JavaFX, Allez dans « File > New Projet… » Et créer un projet « Java with Maven > FXML JavaFX Maven Archetype (Gluon) ». Maven va installer automatiquement JavaFX et mettre en place un projet de base. Vous pouvez ensuite lancer l’application et tout devrait fonctionner.

La raison pour laquelle nous allons utiliser Maven est car l’installation de JavaFX sera faite automatiquement. Additionnement, beaucoup plus de documentation est écrite sur les projets Maven JavaFX que Ant. Notez que l’architecture arm64 n’est pas supportée par Maven - JavaFX. JavaFX n’a pas un bon support de cette architecture et NetBeans causera de nombreuses difficultés durant la mise en place de l’application. Quelle est la situation actuelle ? Vous pouvez éventuellement illustrer le processus.

Vous pouvez aussi remarquer que Maven ajoute 1 fichier important : « module-info.java ». Ce fichier défini le module (l’application) et ses exports, imports, etc…

## Création d'un projet JavaFX

Le test technologique implémenté a pour but de tester la communication entre le Thread Java et le Thread JavaFX. Voici la liste des technologies qui vont être tester :

* Échange de données entre le Thread JavaFX et le Main Thread
* Expérimentation de Platform.runLater
* Expérimentation de Binding
* Expérimentation de Task

Le choix de sa technologie n’a pas été fait au hasard. La communication entre le « backend (Worker) » et le « frontend (JavaFX) » est une des parties les plus complexe de ce projet.

### Structure MVC avec JavaFX

A screenshot of a computer

Description automatically generatedLe projet est nommé JavaFX-TT et sera une application MVC (Model/Worker – View – Controller) simple.

Les Views sont dans le src/main/ressources/app/views au format FXML. Leur location n’est pas aléatoire, « src/main/ressources » est le dosser depuis lequel les ressources de l’application sont copié avec Maven. Vu que les FXML ne sont pas compiler, ce sont des ressources.

Le contrôleur se trouve dans « app/controller » et le Worker se trouve dans « app/workers ».

Figure 1

(Figure 1 : le projet sur NetBeans)

## JavaFX-TT

### Fonctionnement

Quand vous lancez l’application, vous allez voir un champ texte qui est lié à la « value » du label. Une fois que vous appuyez sur « Fetch data », le « value » affiche « Fetching data… » pendant 1 seconde et va charger des valeurs dans le tableau à droite.

Ce test n’est pas très utile mais permet de bien tester le fonctionnement des différentes technologies utile pour le projet

|  |  |
| --- | --- |
| A screenshot of a computer  Description automatically generated | A screenshot of a computer  Description automatically generated |

### Points importants de l’implémentation

Je ne vais pas écrire tout le code de l’application ci-dessous mais uniquement les parties intéressantes.

/\*\*

\* Update the list content

\* @param values The list of values to put in the list

\*/

public void updateList(List<String> values) {

// Use of Platform.runLater because it will be called by an external thread.

Platform.runLater(() -> listTask.getItems().setAll(values));

}

Lors de la mise à jour de la liste de l’application, l’utilisation du Platform.runLater permet à un second thread de modifier le contenu de la liste. Sans ce Platform.runLater, la mise à jour de la liste pourrai créer un java.lang.IllegalStateException. Cette erreur n’apparait pas à chaque clique mais arrivera au bout d’un moment.

/\*\*

\* Bind the label to the input and enable the buttonRunTask.

\*/

private void bindLabelToInputText() {

label.textProperty().bind(textLabel.textProperty());

buttonRunTask.setDisable(false);

}

/\*\*

\* Unbind the label from the input and disable the buttonRunTask.

\*/

private void unbindLabelToInputText() {

label.textProperty().unbind();

buttonRunTask.setDisable(true);

}

Ces 2 fonctions créer/retire le lien entre le texte du label « value » avec le champ texte (textLabel). Sans cette fonctionnalisée, un autre moyen serai d’utiliser un callback ce qui serai beaucoup moins efficient et dure à gérer.

// Create a task to set the new string value

Task<Void> fetchTask = new Task() {

@Override

protected Void call() throws Exception {

// Simulate a long operation (1 second)

wrk.fetchData();

return null;

}

};

// On task success, rebind the text

fetchTask.setOnSucceeded((WorkerStateEvent e) -> {

bindLabelToInputText();

});

// Start the task

new Thread(fetchTask).start();

La création d’une tâche (Task) dans ce cas permet d’ajouter des callbacks pour l’exécution de la tâche. Ici, après que le Worker à récupérer les données, le texte est lié au champ de texte. Comme vous pouvez le voir, la tâche est lancée via la création et exécution d’un nouveau thread.

### Récapitulatifs

* Platform.runLater permet d’interagir avec le thread JavaFX.
* Les Bindings permettent de lier les valeurs avec entre elle.
* Les Task permettent une gestion avancée des Threads et un meilleur contrôle de flux.

## Moving-TT

Ce test technologique a pour but de tester le déplacement d’image sur l’écran. Ceci permet de mieux comprendre comme créer des animations et simulations sur le « backend » et les afficher sur le frontend.

### Fonctionnement

Quand vous lancerez cette application, vous verrez une flèche tourner autour du centre de la fenêtre. Cette application n’a aucune action intéressante mais permet de mieux comprendre comment JavaFX gère ses entités et comment les déplacer sur l’écran.

A screen shot of a computer

Description automatically generated

### Points importants du code

/\*\*

\* Set the arrow position.

\*/

public void setArrowPosition(double x, double y) {

arrow.setLayoutX(x);

arrow.setLayoutY(y);

}

/\*\*

\* Set the arrow rotation in degree.

\*/

public void setArrowRotation(double angle) {

arrow.setRotate(angle);

}

Ceci est le code permettant de déplacer et tourer une ImageView nommée arrow. La modification de ces valeurs ne demande pas de Platform.runLater.

// Update position

// sin(x) = SIN from -1 to 1

// sin(x)+1 = SIN from 0 to 2

// (sin(x)+1)/2 = SIN from 0 to 1

// (sin(x)+1)/2\*n = SIN from 0 to N

double angle = start \* Math.PI \* 2 / SEC\_PER\_ROT;

double xMult = ctrl.getLayerWidth() - ctrl.getArrowWidth();

double yMult = ctrl.getLayerHeight() - ctrl.getArrowHeight();

x = (Math.sin(angle) + 1) / 2 \* xMult;

y = (Math.cos(angle) + 1) / 2 \* yMult;

ctrl.setArrowPosition(x, y);

ctrl.setArrowRotation(-angle / Math.PI / 2 \* 360);

Et voici comment le Worker modifie la position de la flèche. Ce n’est rien de plus qu’un peu de trigonométrie.

// Sleep to reach a FPS goal

double now = getTime();

long sleepMS = Math.round((1.0/FPS - (now - start)) \* 1000);

if (sleepMS > 0) {

try {

Thread.sleep(sleepMS);

} catch (InterruptedException e) {

System.err.println("Unreachable");

}

}

Et voici comment le nombre d’image par second est gérer.

### Récapitulatifs

* ImageView.setLayoutX/Y(double) : Modifie la position d’une image
* ImageView.setRotate(double) : Modifie la direction de l’image en degré