|  |  |
| --- | --- |
|  | **INF-111**  Travail pratique #2  **Groupe : 2 étudiants maximum:**  **Auteur :** Pierre Bélisle (copyright A2022).  **Révision** : Frédérick Simard. |

# Partie 1 de 3

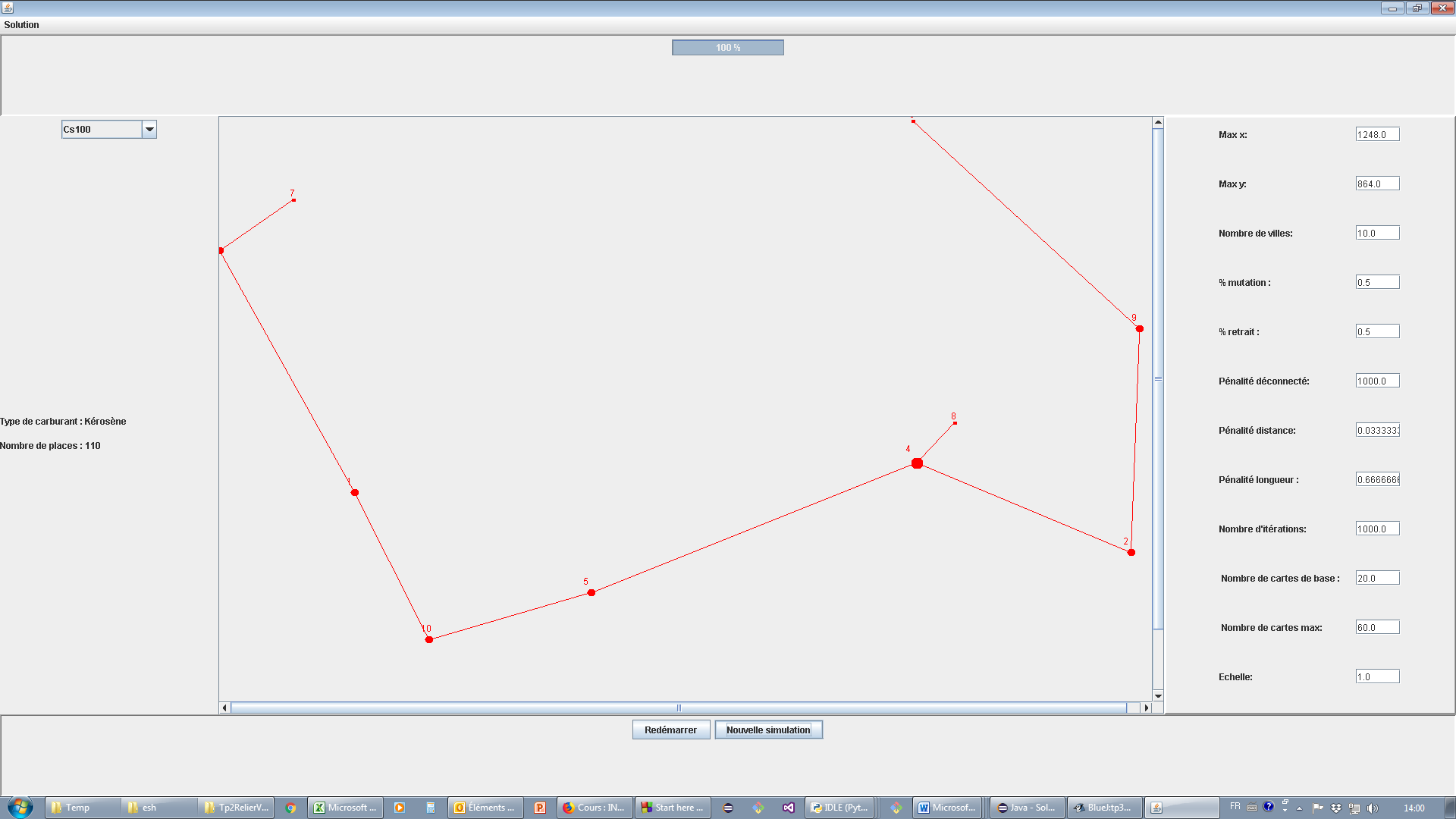
# 1 – Remise en contexte

Pour poursuivre la réalisation du travail sur la liaison de villes du tp2, vous devez maintenant implémenter une version visuelle graphique de l’application. Nous avons besoin de 5 panneaux et un menu. Le panneau du haut montre une barre de progression, le panneau de droite montre les données initiales d’une simulation et en permet la modification. Le panneau du bas montre deux boutons pour réinitialiser ou démarrer une nouvelle simulation, le panneau du centre montre les villes et les liens.

Après que le moteur de calcul a trouvé une solution viable, nous voulons pouvoir simuler les parcours avec différents types de véhicule. Il faudra tenir compte du carburant utilisé et du nombre de passagers maximum que peut contenir le véhicule. L’implémentation de ce panneau est la première partie du travail.

Barre de progression

**Panneau des véhicules**



Barre de menu

Panneau de gestion

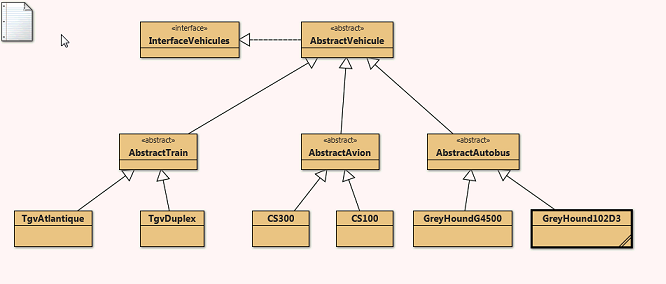
Panneau central

Barre de configuration

# 2- Package hierarchieVehicules: Hiérarchie de classes pour les véhicules.

Il faut pouvoir permettre une sélection de différents types de véhicule soit des avions, (ex : CS100), des trains (ex :TGV Atlantique), et des autobus (GreyHoundG4500). Lorsque l’utilisateur sélectionne un type de véhicule, nous voulons lui montrer le type de carburant et le nombre de passagers maximum du véhicule.

Nous voulons éviter la répétition de code et optimiser l’affichage lors d’un changement de type de véhicule. C’est pourquoi, nous créons une hiérarchie de classes que voici :



Nous vous fournissons la classe **InterfaceVehicules** et un petit programme test. Vous devez écrire le reste des classes que nous vous décrivons ici dans le package hierarchieVehicules.

# 2.1 - AbstractVehicule

Cette classe annonce qu’elle implémente l’interface mais elle ne l’implémente pas au complet. Elle est désignée pour retenir **un** nombre entier qui représente le type de carburant qui peut être un des trois décrits dans la classe **InterfaceVehicule,java.** Vous écrivez le constructeur par copie d’attributs, l’accesseur et le mutateur du type de carburant. Rien d’autres ne sera nécessaire dans le cadre du tp3.

# 2.2 - AbstractAvion, AbstractTrain et AbstractAutobus

Ces classes servent essentiellement à assigner le type de carburant avec le bon carburant. Il n’y a qu’un constructeur par défaut, dans chacune d’elle, qui instancie la classe parent AbstractVehicule avec le bon carburant. Elles ne complètent pas l’interface non plus.

# 2.3 - Les sous-classes de véhicules

Créez une classe pour chacun des véhicules et implémentez la méthode manquante. Sachez que le nombre de passagers pour chacun des véhicules est :

|  |  |
| --- | --- |
| CS100 | 110 |
| CS300 | 130 |
| GreyHound102D3 | 46 |
| GreyHoundG4500 | 55 |
| Tgv Atlantique | 485 |
| Tgv duplex | 455 |

Écrivez aussi la méthode toString qui retourne seulement le modèle du véhicule. (ex : "CS100").

# 2.4 - La classe qui fabrique des véhicules.

Encore pour faciliter le développement, nous vous demandons d’écrire une classe avec une seule méthode ***static*** qui crée et retourne des véhicules selon un paramètre fourni. Cette méthode est utilisée dans le programme test fourni. Il vous indiquera les noms à donner à tout cela.

# 2.5 - Validation

Testez votre hiérarchie et la fabrique avec le fichier **testVehicules.java.**

# 3 – Package vueGraphique : Le cadre.

Créez une fenêtre d’application de type **JFrame** qui implémente l’interface **Runnable,** démarrée dans un programme principal à l’aide de SwingUtilities.invokeLater. Vous trouverez un exemple dans la documentation du cours 10. Vous aurez un complément d’information lors du prochain cours

# 3.1 – Les attributs

Dans votre cadre, vous mettez ces deux attributs. Vous comprenez qu’ils seront instanciés par la méthode **run( )**.

// On adapte à la taille de l'écran qu'on saisit.

**private** Dimension tailleEcran;

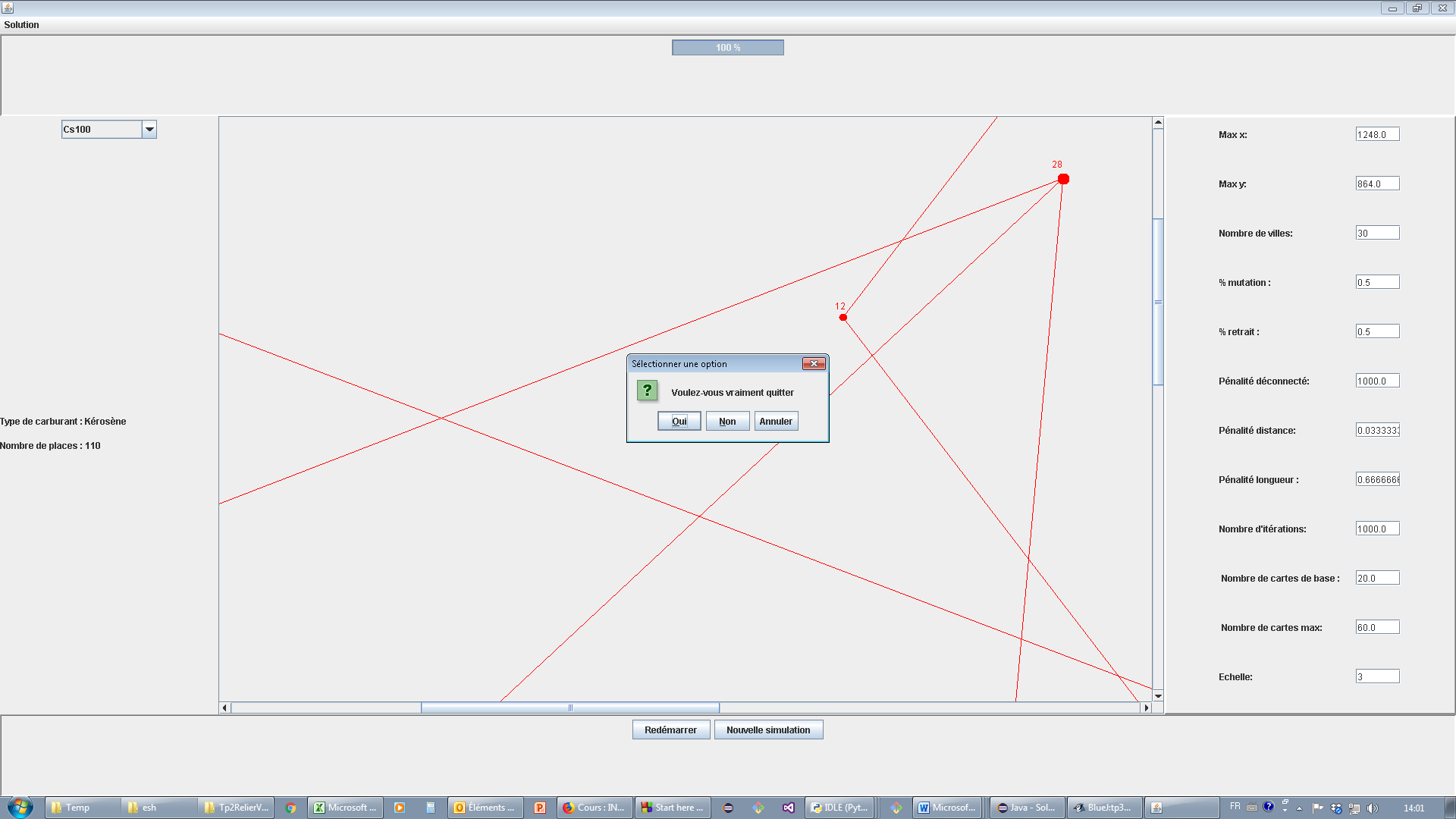
// La configuration est initialisée ici et passé en

// paramètre partout où c'est nécessaire.

**private** Configuration config;

# 3.2 La méthode run()

Dans cette méthode, vous devez obtenir la taille de l’écran[[1]](#footnote-1) et les données initiales de la simulation en instanciant votre attribut config. Ensuite, vous mettez la fenêtre plein écran, vous décidez si vous acceptez que la fenêtre se ferme sur X ou non. Si elle se ferme sur X, vous devez confirmer que l’utilisateur veut bien quitter.



Il reste à remplacer le ***contentPane*** du cadre par une référence vers un objet de la classe PanneauPrincipal définie ci-après et finalement vous mettrez le cadre visible.

**4. Package vueGraphique :** **La classe PanneauPrincipal**

Cette classe vous sera décrite au fur et à mesure du développement. Vous aurez donc à mettre à jour cette classe régulièrement. C’est elle qui conserve les références et dimensionne les différents panneaux.

Elle reçoit la référence des données initiales (celle défini dans le cadre) par son constructeur, et la conserve dans ses attributs.

Pour l’instant, vous y ajoutez en attribut, une instance du panneau de véhicule et une instance du panneau de configuration (définies plus loin). Vous lez dimensionnez et vous les mettez de chaque côté de l’écran (LINE\_START – LINE\_END).

**5. Package vueGraphique :** **La classe PanneauVehicule**

Ce panneau hérite de JPanel et il a un composant pour sélectionner un véhicule et deux autres pour montrer le carburant et le nombre de passagers. Il est possible de le subdiviser en 2 sous panneaux mais ce n’est pas obligatoire.

Le choix des composants Swing est laissé à votre discrétion. L’exemple que nous vous avons donné (**3.2**) utilise un JComboBox et des JLabel. Vous n’êtes pas forcés de les utiliser et nous vous encourageons à en explorer (JList, JTable, …). Le design de l’application n’est pas évalué. C’est dans ce que vous apprenez qui est votre vrai gain.

**6. Package vueGraphique :** **Le panneau de configuration**

Ce panneau hérite de JPanel et il a plusieurs composants pour montrer les données initiales d’une simulation. Les composants utilisés pour l’étiquette et la valeur sont JLabel et JTextField (**3.2** panneau à droite). Vous n’avez pas à le répliquer intégralement. Vous pouvez utiliser les composants et la disposition que vous voulez.

Nous vous suggérons de créer un sous-panneau en classe interne pour regrouper l’étiquette et sa donnée. Cela sera utile dans la partie 2.

**7. Conclusion**

À la fin de la partie 1, votre programme qui teste votre hiérarchie de classe doit fonctionner.

Vous devez être en mesure de voir votre fenêtre d’application avec deux panneaux de côtés opposés.

Pour l’instant aucun événement n’est géré pour les panneaux mais vous pouvez quand même modifier les données initiales à l’écran et sélectionner un véhicule.

**8. Évaluation**

Les mêmes critères de qualité de code et de bonnes pratiques que pour les travaux précédents sont considérés.

Bon travail!

1. (Toolkit.*getDefaultToolkit*().getScreenSize()) [↑](#footnote-ref-1)