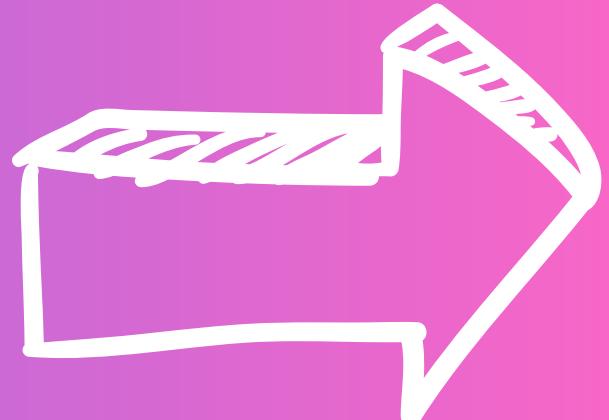
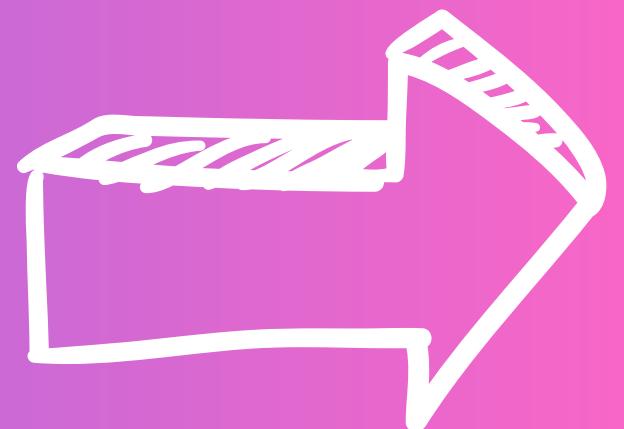


Comment coder un jeu de société ?



Jouer au jeu entre amis pour comprendre les mécaniques

C'est moi



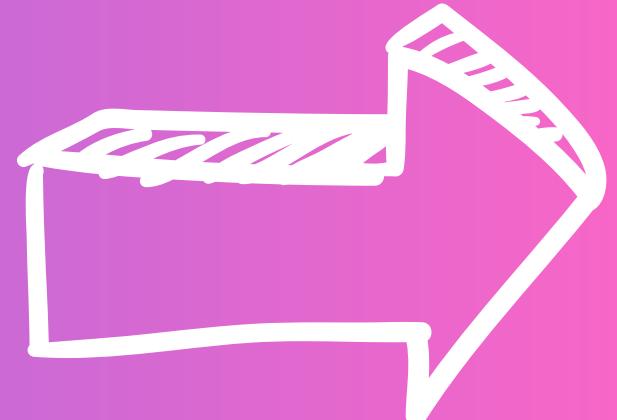
Trouver des failles dans les règles

y a pas de limite de carte ?!

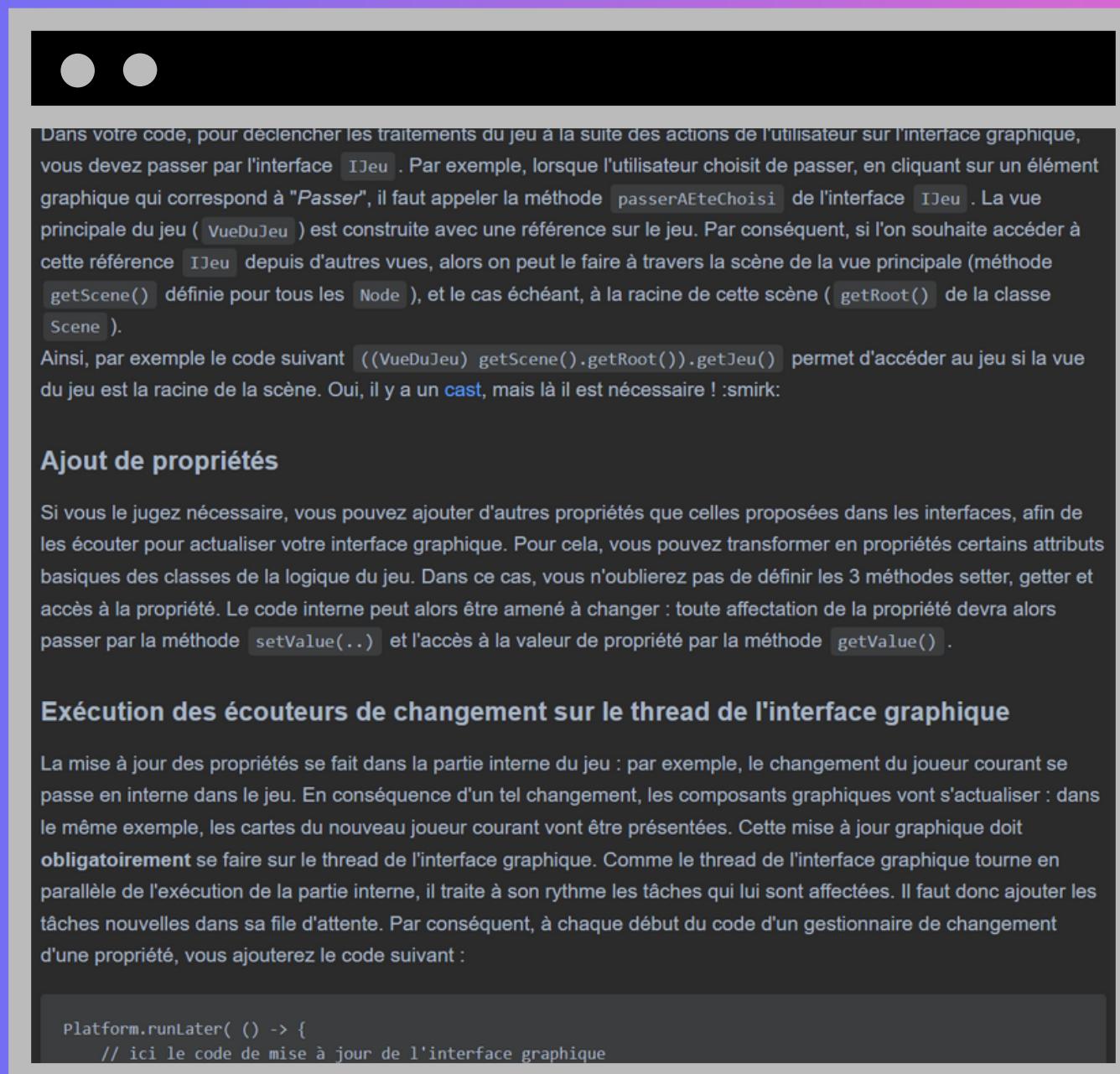
Mais je peux passer tout mes tours en fait



Et m'appeler ""



Comprendre la base du projet



Dans votre code, pour déclencher les traitements du jeu à la suite des actions de l'utilisateur sur l'interface graphique, vous devez passer par l'interface `IJeu`. Par exemple, lorsque l'utilisateur choisit de passer, en cliquant sur un élément graphique qui correspond à "Passer", il faut appeler la méthode `passerAETeChoisi` de l'interface `IJeu`. La vue principale du jeu (`VueDuJeu`) est construite avec une référence sur le jeu. Par conséquent, si l'on souhaite accéder à cette référence `IJeu` depuis d'autres vues, alors on peut le faire à travers la scène de la vue principale (méthode `getScene()` définie pour tous les `Node`), et le cas échéant, à la racine de cette scène (`getRoot()` de la classe `Scene`). Ainsi, par exemple le code suivant `((VueDuJeu) getScene().getRoot()).getJeu()` permet d'accéder au jeu si la vue du jeu est la racine de la scène. Oui, il y a un `cast`, mais là il est nécessaire ! :smirk:

Ajout de propriétés

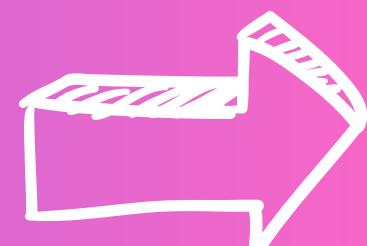
Si vous le jugez nécessaire, vous pouvez ajouter d'autres propriétés que celles proposées dans les interfaces, afin de les écouter pour actualiser votre interface graphique. Pour cela, vous pouvez transformer en propriétés certains attributs basiques des classes de la logique du jeu. Dans ce cas, vous n'oublierez pas de définir les 3 méthodes `setter`, `getter` et accès à la propriété. Le code interne peut alors être amené à changer : toute affectation de la propriété devra alors passer par la méthode `setValue(..)` et l'accès à la valeur de propriété par la méthode `getValue()`.

Exécution des écouteurs de changement sur le thread de l'interface graphique

La mise à jour des propriétés se fait dans la partie interne du jeu : par exemple, le changement du joueur courant se passe en interne dans le jeu. En conséquence d'un tel changement, les composants graphiques vont s'actualiser : dans le même exemple, les cartes du nouveau joueur courant vont être présentées. Cette mise à jour graphique doit obligatoirement se faire sur le thread de l'interface graphique. Comme le thread de l'interface graphique tourne en parallèle de l'exécution de la partie interne, il traite à son rythme les tâches qui lui sont affectées. Il faut donc ajouter les tâches nouvelles dans sa file d'attente. Par conséquent, à chaque début du code d'un gestionnaire de changement d'une propriété, vous ajouterez le code suivant :

```
Platform.runLater( () -> {
    // ici le code de mise à jour de l'interface graphique
```

Comprendre l'**architecture** du code en lisant la **documentation** du projet.



Coder

```
    }

    return verification;
}

/**
 * Trouve la couleur(hors loco) des cartesWagons avec une longueur donnée
 */
3 usages  ↗ Lunalaaaa +1
public CouleurWagon trouveCouleurWagonQuandGris(Joueur j) {
    ArrayList<CouleurWagon> ord = new ArrayList<>(j.getCartesWagon());
    CouleurWagon coul = ord.get(0);
    Collections.sort(ord);
    int max = 1;
    CouleurWagon coulDef = ord.get(0);
    int cpt = 1;
```

Rédiger toutes les fonctions demandées
(avec 310* lignes de code dans chaque classe)

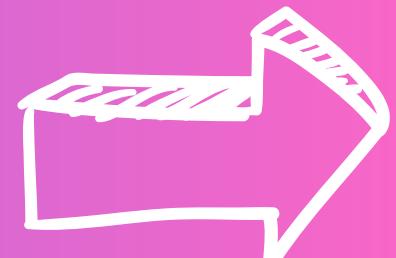


*Nombre réel

Tester

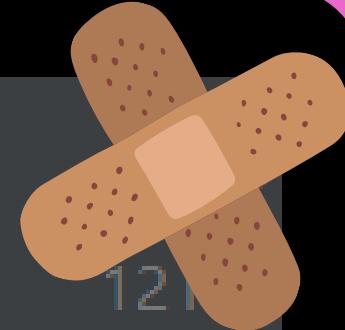
JoueurTest (fr.umontpellier.iut.rails)	307 ms
🚫 testPoserGare()	1 ms
🚫 testCarteValide()	
🚫 testPoserTunnel()	
✓ testVerifierTrajetGris()	27 ms
✓ testVerifierTrajetLoco()	2 ms
✓ testJouerTourCapturerFerryLongueur3()	106 ms
✓ testVerifierTrajet()	1 ms
✓ testVerifierTunnel()	1 ms
❗ jouerTourConstruireQuatreGares()	51 ms
✓ testVerifierTunnelFalse()	3 ms
✓ testJouerTourCapturerFerryLongueur2Loco()	17 ms
✗ jouerTourConstruireTroisGares()	31 ms
❗ jouerTourConstruireDeuxGares()	22 ms
🚫 testCartesTri()	
✓ testVerifierFerry()	1 ms
✓ testVerifierTrain()	1 ms

Apparemment ça ne suffit pas pour que les tests passent en une seule fois.

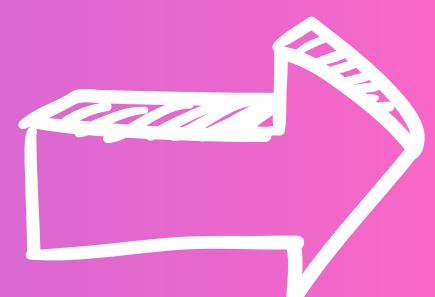


Corriger le code

✓ testCartesPoseesTriCouleur()	12 ms
✓ jouerTourConstruireGaresLoco()	4 ms
✓ testChoisirDestinations()	2 ms
✓ testJouerTourPrendreCartesWagon()	4 ms
✓ testJouerTourCapturerFerry()	6 ms
✓ testJouerTourCapturerRoute()	7 ms
✓ testVerifierTrainFalse()	1 ms

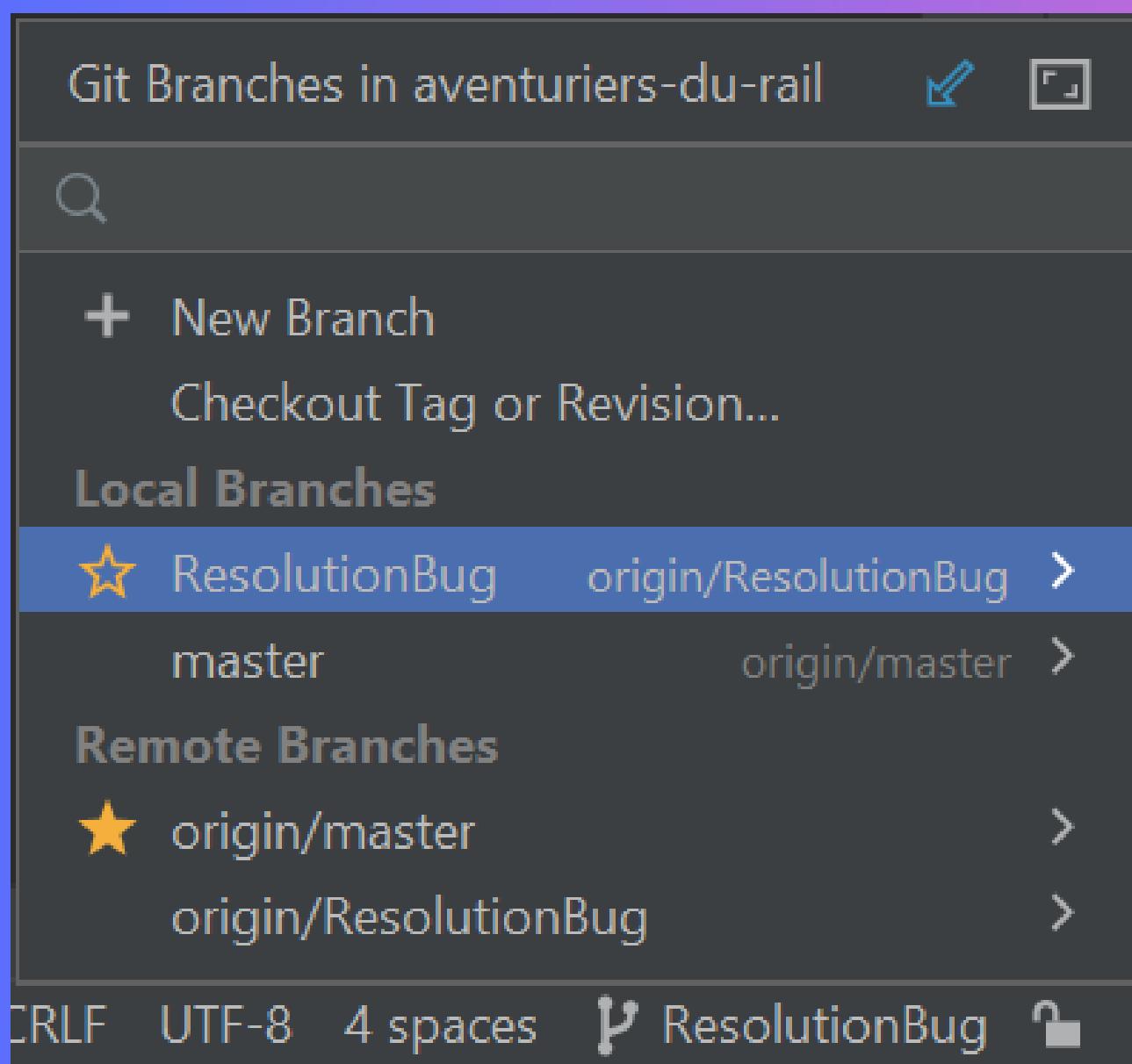


Un code testé est un code mieux pensé



Vouloir envoyer son code

Oups... Pas sur la bonne
branche



Faire les bons merges

Accept Left

Accept Right

Choisir les bons changements de code
en ne perdant aucun morceau de code



Merge remote-tracking branch 'origin/ResolutionBug' into ResolutionBug



origin & ResolutionBug Lunalaaaa

10/04/2022 21:26

