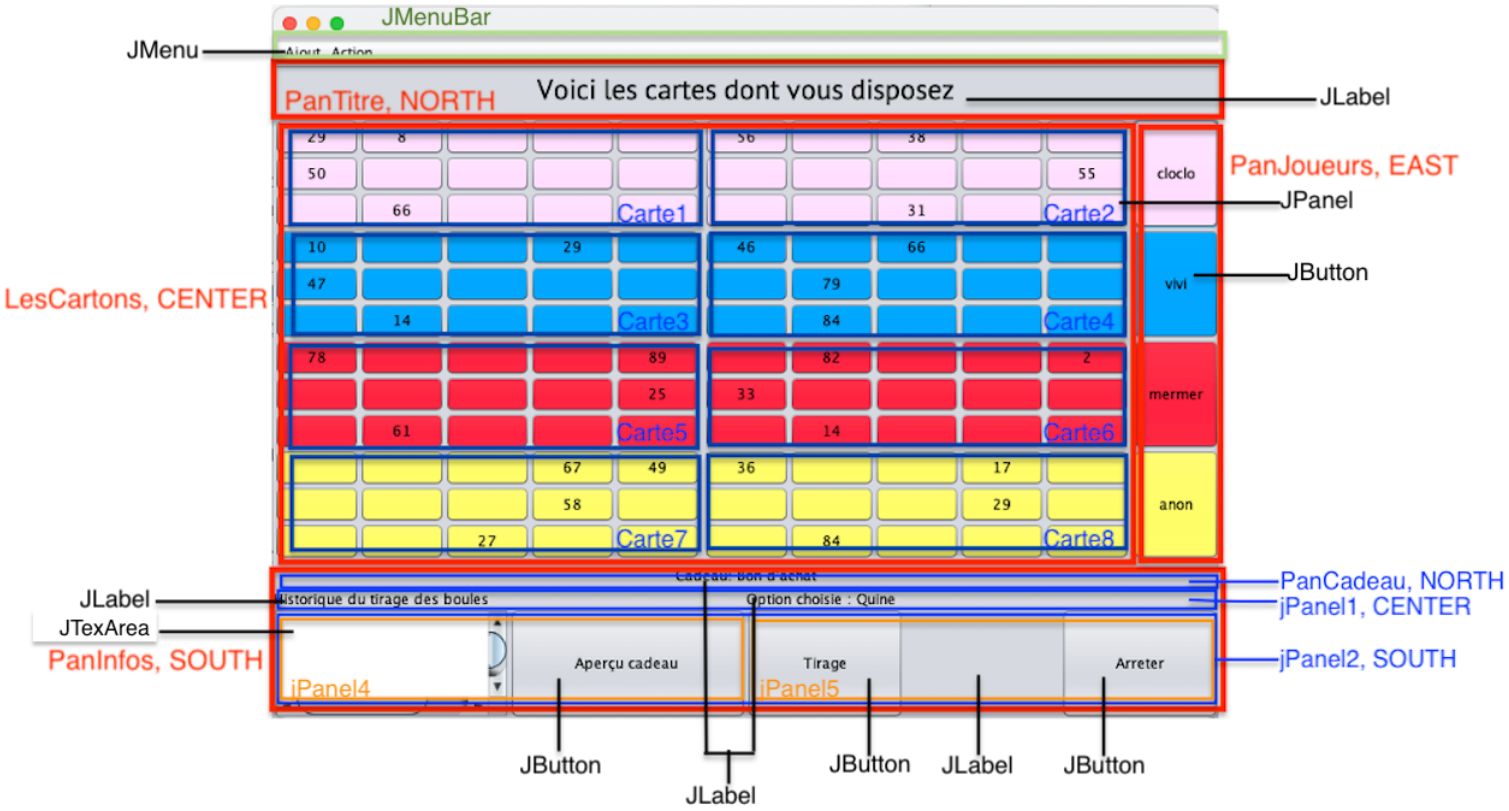
Rapport projet : Le Jeu Loto

Arborescence de la fenêtre de l'application principale



LeJeuLoto JFrame

JMenuBar jMenuBar1

JMenu MAjout « Ajout »

JMenuItem ItemJoueur « Joueur »

JMenuItem ItemLot « Lot »

JMenu MAction « Action »

JMenuItem ItemOption « Option »

JMenuItem ItemDemarrer « Démarrer Jeu »

JMenuItem ItemAcheter « Acheter cartes »

JMenuItem ItemContinuer « Continuer »

JMenuItem ItemRecommencer « Recommencer »

BorderLayout

JPanel PanTitre, NORTH
FlowLayout
JLabel LTitre « Voici les cartes dont vous diposez »
JPanel LesCartons, CENTER
GridLayout(4,2)
JPanel Carte1
(*par défaut FlowLayout)*
JPanel Carte2
(*par défaut FlowLayout)*
JPanel Carte3
(*par défaut FlowLayout)*
JPanel Carte4
(*par défaut FlowLayout)*
JPanel Carte5
(*par défaut FlowLayout)*
JPanel Carte6
(*par défaut FlowLayout)*
JPanel Carte7
(*par défaut FlowLayout)*
JPanel Carte8
(*par défaut FlowLayout)*
JPanel PanJoueurs, EAST
GridLayout(4,1)
JButton BJoueurs1 « Joueur 1 »
JButton BJoueurs2 « Joueur 2 »
JButton BJoueurs3 « Joueur 3 »
JButton BJoueurs4 « Joueur 4 »
JPanel PanInfos, SOUTH
GridLayout(4,1)
JPanel PanCadeau, NORTH
FlowLayout
JLabel LCadeau « »
JPanel jPanel1, CENTER
GridLayout(1,2)
JLabel jLabel3 « Historique du tirage des boules »
JLabel LOption « Option choisie : »
JPanel jPanel2, SOUTH
GridLayout(1,2)
JPanel jPanel4
GridLayout(1,2)
JTextArea HistoTirage
JButton BCadeau « Aperçu cadeau »
JPanel jPanel3
GridLayout(1,3)
JButton BTirage « Tirage »
JLabel jLabel3 « »
JButton BArreter « Arreter »

Les attributs et le constructeur de la classe LeJeuLoto

LeJeuLoto est une sous-classe d'une JFrame, elle est la classe principale de notre application. Elle dispose de dix attributs qui stockent les différents éléments dont le jeu a besoin pour fonctionner.
Certains sont destinés à sauvegarder les dimensions de notre carte. Ce sont des entiers appelés nbCol (pour le nombre de colonne d'une carte) et nbNum (pour le nombre de numéros dans une carte de loto). Ils sont initialisés à 5 pour le nombre de colonnes et 4 pour le nombre de numéros.
L'attribut option est un entier qui peut être 1, 2 ou 3 et à chacune de ces valeurs est associée une option pour gagner le jeu. Dans le constructeur de la classe, on l'initialise à 1 et on affiche un message grâce au label LOption.
Ensuite deux, attributs de liste de joueurs de type LesJoueurs sont nécessaires. Une pour stocker les joueurs du jeu que l'on nommera lstJ et l'autre gagnants pour stocker les gagnants de la partie. On les initialise de la même façon dans le constructeur, c'est-à-dire en créant un objet de type LesJoueurs.
Il faut après gérer les lots, et le lot courant attribué pendant la partie. Nous avons donc un attribut lstL de type LesLots que l'on déclare dans le constructeur en créant un objet. Puis il y l'attribut lc de type Lot que l'on déclare null dans un premier temps. Pour remplir notre liste de lots lstL on appelle une méthode initLots() dans notre constructeur. Cette méthode crée un objet l1 (par exemple) de type Lot.
L'attribut carteAchetee est un boolean qui tout au long de la partie va nous dire si des cartes de loto ont été achetées par les joueurs. Dans le constructeur il est initialisé à false car aucune carte n'a été achetée.
Pour finir, les attributs de la gestion des boules sont, un tableau a une dimension d'entiers appelés boules puis un entiers MAX qui est la taille maximale du tableau et il est fixé à 90 et sa valeur est non modifiable. Le tableau boules est déclaré dans le constructeur avec en paramètre MAX, le constructeur appelle une méthode initBoules() qui va remplir notre tableau de 0.
Puis la méthode initComponents() qui construit notre interface.

Strainchamps Clothilde MI-I12

Les actions à réaliser avant le démarrage du jeu

Pour qu'une partie de loto puisse démarrer plusieurs éléments sont requis :

1. Il doit y avoir au moins un joueur dans la partie, pour cela l'utilisateur doit sélectionner dans la barre de menu le menu ajout et ensuite l'option « Joueur ».
   En cliquant dessus cela va ouvrir une boîte de dialogue pour ajouter un joueur.
   L'utilisateur pourra alors entrer le pseudo du joueur, lui choisir une couleur, de plus un joueur possède un solde de 20 euros au début de la partie. Ensuite pour valider sa sélection l'utilisateur doit cliquer sur le bouton ''Valider''. Ensuite l'utilisateur voit son ajout sur les boutons de joueur avec le pseudo et la couleur qu'il a choisi pour son joueur.
   On peut ajouter jusqu'à quatre joueurs maximums dans la partie.
   ⇒ (détailler dans le pré-rapport)

2. Ensuite chaque joueur doit avoir acheté soit une carte minimum ou deux cartes de loto maximum. L'utilisateur doit alors dans la barre de menu sélectionner ''Action'' puis « Acheter cartes », cela va ouvrir une boîte de dialogue pour acheter des cartes de loto.
   Dans cette boîte de dialogue une carte est affichée. On sélectionne grâce à une liste déroulante le joueur qui veut acheter une carte, si la carte présentée convient au joueur il peut alors l'acheter en appuyant sur le bouton « Acheter » ou bien il peut appuyer sur le bouton « Autre choix » et avoir une nouvelle carte affichée.
   Une fois tous les achats de cartes réalisés l'utilisateur clique sur le bouton « Quitter » pour revenir à la fenêtre principale.
   ⇒ Pour cela il nous faut créer une boîte de dialog JDialog que l'on nommera « AchatCarteDlg » qui dérive de notre JFrame « LeJeuLoto ».
   ⇒ Notre classe AchatCarteDlg dispose de 8 attributs. Nous avons les attributs qui gèrent la taille de la carte donc le nombre de lignes nbLig et le nombre de colonnes nbCol puis le nombre de numéros dans la carte nbNum.
   Nous avons également la liste courante de nos joueurs ljc et le joueur courant jc sélectionné grâce à une comboBox, la carte courante affichée dynamiquement par le programme.
   Il y a aussi deux boolean, un pour savoir si un achat a été réalisé et un indicateur de fermeture pour la boîte de dialogue.
   ⇒ Cette classe reçoit des informations de la classe principale LeJeuLoto, comme les joueurs, les lignes/colonnes de la carte puis le nombre de numéros, on les stocke dans les différents attributs.
   ⇒ À l'ouverture de la boîte de dialogue une carte est générée, (on utilise pour cela les propriétés d'une carte loto pour donner les numéros de la carte entre 1 et 90 et à des emplacements sur la carte aléatoire. Ensuite il faut afficher cette carte pour cela deux méthodes sont nécessaires. La première initCarte() va générer dans l'interface les boutons représentant les cases de notre carte, puis la deuxième est afficheCarte() qui dans les boutons va afficher dans les boutons les numéros de la carte aux bons emplaçements.

- Dans les méthodes initCarte() et afficheCarte() on se sert des attributs nbLigne et nbColonne pour les dimensions (on fixe un GridLyout en fonction) de la carte à afficher.

⇒ Au clic sur le bouton ''Autre choix'' cela va effacer totalement notre panneau où la carte est générée (donc les boutons) puis créer une nouvelle carte courante stockée dans l'attribut cc, on l'initialise et on l'affiche.

- Pour cela une méthode BAutreChoixActionPerformed(..) est écrite.

⇒ En cliquant sur le bouton ''Acheter'' le joueur sélectionné (jc) dans la comboBox va acheter la carte courante affichée. Cette action va appeler la méthode acheter(carte) de la classe joueur (méthode renvoyant un boolean) pour savoir si l'achat est possible. Si c'est le cas on affiche le nouveau solde du joueur dans un label LInfos, l'attribut achat devient alors true, puis on efface notre panneau, on génère une nouvelle carte et on l'affiche... Si l'achat n'est pas possible on affiche alors dans LInfos la raison de l'impossibilité de cet achat.

- Pour cela une méthode BAutreChoixActionPerformed(..) est écrite.

Pour finir, en cliquant sur le bouton ''Quitter'' l'utilisateur va valider tous les achats de carte des joueurs. La boîte de dialogue d'achat des cartes va alors renvoyer un boolean à la classe principale LeJeuLoto pour savoir si des cartes ont été achetées, on le stockera dans l'attribut carteAchetee de LeJeuLoto et les cartes achetées (cette information est importante pour le choix des options car toutes les cartes du jeu doivent avoir les mêmes dimensions) seront affichées en face de chaque joueur grâce à une méthode afficheCartes() écrite dans la classe principale LeJeuLoto qui sera décrite plus tard dans le rapport.
Cette information est importante pour le choix des options car toutes les cartes du jeu doivent avoir les mêmes dimensions.

- Pour cela une méthode BAutreChoixActionPerformed(..) est écrite.
- De plus une méthode ItemAcheterActionPerformed(...) est écrite dans LeJeuLoto pour réaliser ces échanges.

3. L'utilisateur peut choisir d'autres paramètres de jeu que ceux prédéfinis. Il doit alors dans la barre de menu sélectionner « Action » puis « Option Jeu », cela va ouvrir une boîte de dialogue pour choisir les différentes options de jeu.
Il peut alors choisir combien de lignes complètes il faut pour gagner le jeu (Quine, Double Quine, Carton Plein) en sélectionnant l'un des boutons radio, il peut également choisir le nombre de colonnes qu'une carte peut avoir grâce à une liste déroulante et également le nombre de numéros que vont contenir les cartes de loto dans une zone de texte.
Si cette boîte de dialogue est ouverte avant l'achat des cartes de loto alors toutes les options pourrons être choisies.
Sinon, seule l'option de coup gagnant pourra être modifiée car dans le jeu toutes les cartes doivent avoir le même nombre de lignes, colonnes et de numéros.
Une fois les différentes options choisies l'utilisateur peut appuyer sur le bouton « Valider » ce qui enregistrera ces choix et la fenêtre se fermera, ou bien il peut cliquer sur « Annuler » ce qui fermera la fenêtre sans enregistrer ces choix.

⇒ Pour cela il nous faut créer une boîte de dialogue JDialog que l'on nommera « OptionsDlg » qui dérive de notre JFrame « LeJeuLoto ».

⇒ Notre classe OptionsDlg dispose de 4 attributs. Nous avons les attributs qui gèrent la taille de la carte donc le nombre de colonne nbCol et le nombre de numéros dans la carte nbNum.
Nous avons un attribut choixOpt pour stocker la valeur de l'option choisie en fonction de quel bouton radio est sélectionné. Trois valeurs sont alors disponibles 1, 2 ou 3.
Puis un boolean comme indicateur de fermeture pour la boîte de dialogue.

⇒ Cette classe reçoit des informations de la classe principale LeJeuLoto, comme les joueurs, les colonnes de la carte, le nombre de numéros, on les stocke dans les différents attributs.
Un boolean pour savoir si des achats de cartes ont été réalisés auparavant d'où l'importance du boolean que renvoie la classe AchatCarteDlg. Si aucun achat n'a été fait on fixe des valeurs par défaut pour les différents attributs qui sont modifiables grâce à la comboBox et au textField et on stocke les nouvelles valeurs. Si achat alors ces caractéristiques sont non modifiables et la comboBox et le textField sont inutilisables.

- Une méthode remplirComboCol() est alors écrite pour remplir la comboBox.

⇒ Au clic sur le bouton ''Valider'', tous ces choix vont être enregistrés dans les différents attributs. La boîte de dialogue des options va alors les envoyer à la classe principale qui les stockera dans ces attributs nbCol, NbNum et option.
De plus en fonction du choix de l'option le label LOption changera pour indiquer le choix dans l'interface.

- Pour cela une méthode BValiderActionPerformed (..) est écrite.
- Et une méthode ItemOptionActionPerformed (...) est écrite dans LeJeuLoto pour réaliser ces échanges.

4. L'utilisateur peut visualiser chaque joueur en cliquant sur les boutons respectifs. Une boîte de dialogue s'ouvre alors. Il peut alors voir le pseudo du joueur, sa couleur, son solde, s'il a gagné des lots et une photo par défaut.
   ⇒ (détailler dans le pré-rapport)

Les actions à effectuer ou possibles pendant le jeu

1. Une fois que toutes les données requises pour démarrer le jeu sont entrées, le jeu peut alors commencer. L'utilisateur peut alors sélectionner dans la barre de menu ''Action'' puis ''Démarrer Jeu''. Cela va alors débloquer le bouton tirage, sélectionner un lot aléatoirement et l'afficher sur un bouton et sa description dans un label, dans l'interface principale.
   ⇒ Pour cela nous avons besoin de créer une méthode partiePrete() qui vérifie que tous les éléments nécessaires au démarrage d'une partie sont présents, sinon elle nous indique dans le label LTirage que la partie n'est pas prête.
      - Une méthode ItemDemarrerActionPerformed(...) est alors écrite dans la classe LeJeuLoto, elle est appelée au clic sur l'item ''Démarrer Jeu''.

2. En appuyant sur le bouton tirage, des boules seront tirées aléatoirement, elles portent un numéro aléatoire en 1 et 90 qui ne peuvent être tirés qu'une seule fois. L'utilisateur pourra voir sur l'interface principale quel est le numéro de la boule tiré car il sera affiché dans un label. L'historique des tirages sera également affiché dans un textArea. Au moment du tirage si le numéro de la boule tiré correspond à un numéro sur une des cartes de loto des joueurs alors un jeton est posé sur la carte.
   Ensuite si un joueur gagne en fonction de l'option choisie, il est affiché dans le textArea et le jeu s'arrête. Deux choix s'offrent alors à l'utilisateur : il peut soit continuer la partie, ou bien recommencer la partie de zéro.
   ⇒ Pour le tirage d'une boule, on génère un nombre aléatoire nb entre 1 et 90, de plus pour que ce numéro ne soit tiré qu'une seule fois on crée un tableau de taille 90, initialisé à zéro dans toutes ses cases (grâce à une méthode initBoules()) et quand une boule est tirée on stocke 1 dans la case nb-1 du tableau la valeur 1. Ensuite nb est affiché dans LNombreTirage dans l'interface, et ajouté à HistoTirage.
   ⇒ À chaque nouveau tirage, pour chaque joueur du jeu, on regarde chacune de ses cartes et on place un pion sur sa carte avec la méthode placePion(nb).
   ⇒ Puis pour voir s'il y a un gagnant on applique la méthode cartonGagnant(option), si cette méthode nous renvoie true alors ce joueur est ajouté à notre attribut gagnants qui correspond aux gagnants du jeu, puis on affiche en fonction de l'option que le joueur a réalisée un Quine, par exemple. Si deux joueurs sont gagnants au même tirage, on pioche un gagnant du lot au hasard entre eux.
   ⇒ A chaque fois on réaffiche également les cartes des joueurs grâce à la méthode afficheCartes(). Dans cette méthode on initialise un compteur à 0 et pour chaque joueur, on regarde le nombre de cartes du joueur, s'il n'en a pas on ajoute deux à la variable compt (par exemple pour libérés les panneaux Carte1 et Carte2 qui ont un indice de composant 0 et 1 dans notre interface, si le joueur à une carte on dessine la carte grâce à la méthode dessineCarte() implémenter dans la classe Carte, d'indice 0 dans le panneau d'indice compt, ainsi de suite pour deux cartes...
      - Une méthode BTirageActionPerformed(...) est alors écrite dans LeJeuLoto, et est appelée au moment où l'utilisateur clique sur le bouton.

- Une méthode afficheCarte() est écrite dans LeJeuLoto.

3. Pendant le jeu l'utilisateur peut toujours visualiser les joueurs.

4. En revanche aucun ajout de joueur ou achat de carte n'est possible pendant le jeu.

Les actions pour poursuivre le jeu le cas échéant

1. Pour poursuivre le jeu, l'utilisateur peut décider de continuer la partie en cours, pour cela il peut alors sélectionner dans la barre de menu ''Action'' puis ''Continuer''.
   Cela va alors monter l'option de jeu de un (par exemple un Quine deviendra un Double Quine...) si l'option CartonPlein a été sélectionnée au début la partie alors le jeu ne pourra pas continuer. Le bouton tirage redevient alors cliquable et un nouveau lot est affiché dans l'interface comme au lancement de la partie.
   De plus un nouveau lot courant sera sélectionné au hasard.
   ⇒ On ajoute donc un à l'attribut option de la classe LeJeuLoto si option était égal à 1 ou 2 et on modifie en conséquence le label LOption sinon on affiche que le niveau de carton est atteint dans LOption.
   ⇒ On génère un nombre aléatoire pour choisir notre lot dans notre liste de lots, on l'ajoute à l'attribut lc qui est notre lot courant, l'affiche sur notre bouton BCadeau et sa description dans le label LCadeau.
   ⇒ La liste des gagnants est vidée grâce à la méthode supprimerJoueur() de la classe LesJoueurs.
      - Une méthode ItemContinuerActionPerformed (...) est alors écrite dans LeJeuLoto et qui est appelée quand l'utilisateur sélectionne l'item dans le menu.

2. Dans un autre cas l'utilisateur peut dans la barre de menu sélectionner ''Ajout'' puis ''Recommencer''.
   L'historique des tirages est alors vidé et toutes les configurations du départ sont restaurées. En ce qui concerne les joueurs, leurs soldes sont remis à 20€ et leurs anciennes cartes sont effacées de leurs liste de carte et de l'interface. Les boutons ''Tirage'' et ''Arrêter'' ne sont plus cliquables.
   Si le joueur veut redémarrer une partie, il doit acheter des cartes... pour que la partie soit prête.
   ⇒ les attributs option et carteAchetee reprennent respectivement 1 et false comme au lancement de l'application.
   ⇒ Pour chaque joueur on modifie son solde avec la méthode setSolde(s) de la classe Joueur puis on efface les cartes de la liste des cartes achetées avec la méthode retireCarte() de la classe LesCartes.
   ⇒ Pour effacer les cartes des joueurs de l'interface principale on applique la méthode removeAll() sur les panneaux de cartes (de Carte1 à Carte8).
   ⇒ La visibilité des boutons et items est remise comme au démarage, en somme tout est remis à zéro sauf la liste des joueurs.
   ⇒ La liste des gagnants est vidée grâce à la méthode supprimerJoueur() de la classe LesJoueurs.
      - Une méthode ItemRecommencerActionPerformed (...) est alors écrite dans LeJeuLoto et qui est appelée quand l'utilisateur sélectionne l'item dans le menu.

Strainchamps Clothilde MI-I12

Les actions liées à l'arrêt du jeu

1. Pour commencer l'arrêt est causé par le fait qu'un joueur soit déclaré gagnant dans la méthode sur le bouton tirage comme expliqué précédemment.

2. L'utilisateur peut également décider de lui-même d'arrêter la partie de jeu en cliquant sur le bouton ''Arrêter''. Cela va alors rendre le bouton ''Tirage'' non cliquable, et il devra alors sélectionner l'item ''Recommencer'' dans le menu (action décrite ci-dessus).
   ⇒ Pour cela on doit rendre les boutons ou item visibles en fonction de nos souhaits cliquables ou non.
   - Une méthode BArreterActionPerformed (...) est alors écrite dans LeJeuLoto et qui est appelée quand l'utilisateur clique sur ''arrêter''.

Le projet sera considéré comme réussi si l'ensemble des conditions décrites ci-dessus sont respectées.