INTRODUCTION :

Notre projet EUREKA est un jeu de pendu classique basé sur la culture générale qui s'intéresse a poser des question au joueurs , les question seront divisée en deux catégorie (enfant et adulte selon l'âge) dans cinq différents thème . le principe est de poser une question a l'utilisateur et c'est à lui de trouver la bonne réponse en entrant des caractere dans une zone de saisie , chaque a 8 chances pour trouver la bonne réponse , si il trouve la reponse avant de terminer ses chances il sera considéré comme gagnant de la partie donc le niveau . chaque thème est composé de 5 niveau

DÉROULEMENT :

En lançant le jeu , il va automatiquement charger la liste des question et le Hashmap des joueurs de fichier Joueurs.txt et Question.txt grace a les methodes définie dans la classe ReadWrite

Tout d'abord le joueur commence par s'inscrire donc il doit tout d'abord son nom prénom date de naissance et un mot de passe , les nouveau joueur inscrit seront classé en Adulte ou Enfant grâce à leur date de naissance en il va etre ajouter dans le Hashmap des joueur avec une liste des dernier niveau par thème ou chaque niveau est a 0 est on crée une nouvelle liste qui contient les partie de jeux realiser , le programme va lui donner un nom d'utilisateur sous la forme prenom.numero-sequentiel pour l'utiliser quand il veut jouer ,sinon si il a un compte il saisit son nom d'utilisateur et son mot de passe dans la fenêtre Bienvenue

Après l'inscription ou la connexion , le programme charge le hashmap des thème avec des listes de question (linkedlist) selon la catégorie de joueur courant donc la liste des question (linkedlist) des thème elle contient des joueur de type QuestionAdulte ou bien QuestionEnfant et non pas un mélange des deux

Sur l'écran une fenêtre thème qui s'affiche ou le joueur doit choisir un thème de la liste des thème (si il a deja finie un thème ce dernier sera invisible) pour commencer sa partie , après la fenêtre de jeu (PENDU) s'affiche ou elle contient une question une zone pour saisir les caractère une zone pour afficher l'image pendu et le score et un bouton retour a la fenêtre des thème dès que le joueur entre son premier caractère le button retour sera invisible et la partie commence et elle elle sera enregistrée dans la liste des partie jouer

Dans chaque theme le joueur va avoir 5 niveau , pour un joueur adulte il aura 2 chance par niveau et pour l'enfant 3 sinon il perds le niveau donc le jeu dans ce cas la le score va retourner a 0 et la liste des dernier niveau se réinitialise a 0 pour chaque thème

ARCHITECTURE :

1) pour l’interface graphique on a utilisé une fenetre Fenetre de type JFrame et 5 panel Bienvenue SingUp Pendu Theme et Potence , le principe et d’afficher les panel dans la Fenetre mere , cette Fenetre contient un menu bar avec 4 buttons (Apropos , Help ,Players et Quitter pour quitter le jeux) dans le bouton quitter il y a une action qui enregistre le fichier des joueurs avec les dernière modification avant de quitter avec la désactivation de bouton fermer standard . Pour le basculement des panel on met chaque panel dans un panel vide dans la fenetre grace a la fonction refreshPanl(JPanel p).
2) pour les fichier : dans le jeux on a utilisé deux fichier le premier est un fichier qui un hashmap des joueurs (un fichier dynamique) et le deuxième et un fichier texte qui contient des question , la forme des questions dans les fichier est :
   Catégorie:NUMqst:question:réponse:lien d’image : niveau ;
   si la question elle contient pas d’image on mets rien , donc notre fonction de readQuestion() elle décompose les question par ligne en enlevant les ;de la fin et les : , après elle mets les différent composant de chaque ligne dans une tableau de type String , si la 1er case (catégorie) est de type enfant on crée un objet question de type questionEnfant et on l’ajoute dans la liste des question , sinon on crée un objet de type questionAdulte , et pour le hashmap des joueur c'est la méthode readHashMap() qui recupere le hashmap de fichier qui contient la liste des joueurs
3) pour l’inscription ou la connection : le joueur doit saisir son nom d’utilisateur et mot dans pas dans les zone de panel Bienvenue si les information sont juste la fonction SeConnecter(user name, mot passe) va afficher le panel de Thème ou le joueur va choisir son thème et elle charge le hashmap des question avec une liste de question de même type que le joueur inscrit grace a la methode chargerThemeHashset(etat,ArrayList<Question> qui contient la liste de tous les question) appele dans le corp de la methode Seconnecter() sinon il doit inscrire dans le panel SingUp ou il entre son nom prénom mot de passe et date de naissance le programme va créer un nouveau objet de type Adulte ou Enfant selon son âge grâce a la méthode Inscription() , cette méthode va créer un nom d'utilisateur qui est composé de prénom de joueur + un numéro séquentiel (le nombre de joueur dans le hashmap +1) et l’affiche dans le panel Sign Up , elle appelle aussi la méthode qui crée la liste des thème avec les question
4) pour les thème et le score affiché dans la panel thème : le score afficher thème est calculé grâce a la methode get Total Score() qui calcule le score de joueur courant , pour chaque thème on a coif pour cela si dans sa liste de dernierNiveau par exemple le thème Santé est a 5 il aura le score de les question de 5 niveau cumulé * le coif associé au thème santé.Pour la liste des thème on crée une liste en l’ajoutant les thème de jeu pour cela on doit vérifier si le joueur a complété le thème donc on vérifie la liste des dernier

niveau pour chaque thème . après le choix de thème il appuie sur le bouton commencer

5) le panel pendu contient une image de notre pendu une question avec ou sans image , une zone pour saisir le caractère et un textfield qui contient une réponse en *** grâce a la méthode PenduInitialisationRep() , la question associé a la partie est créé grâce à la méthode initialiserPartie() qui cree une question et une nouvelle partie , la question obtenu est grace a la methode Jouer de Joueur inscrit , elle nous rends la question actuelle pour le niveau et elle initialise les élément de ce panel grâce à la méthode setGame() . En tapant le premier caractère dans la zone le bouton Retour va disparaître la partie sera enregistré dans la liste des partie jouer . En tapant des caractère le caractère va être envoyé à la classe Partie jeu ou elle contient une méthode checkCaractère() pour vérifier le caractère , dans on a 2 cas

    1: le caractère est juste donc elle remplace l'* correspondante Au char par ce caractere et elle doit vérifier si le mot ne contient pas des *, dans ce cas la elle increment le niveau en vérifiant si le niveau actuel est différent de 5 , sinon elle va revenir au panel Thème

    2: dans le cas contraire elle va incrémenter l'état de notre image pendu si l'etat ==8 la partie est perdu donc elle va vérifie le nombre des partie perdu pour ce niveau si il égale a notre nombre d'essaie associé au joueur (2 pour adulte et 1 pour enfant) (le nombre est le 5eme caractère dans le ID de la question exemple HIS32 le nombre de la question est 2), dans le cas ou les 2 nombres sont égaux le joueur est considéré comme perdant et elle appelle la méthode restartJoueur() pour réinitialiser le joueur sinon pour réinitialiser la partie

la méthode Joueur de la classe Joueur : elle nous rend la prochaine question de thème en vérifiant le dernier niveau et si la question est déjà joueur
pour cela on parcourt la liste des question de thème donné (on obtient la liste grâce à la méthode getQuestions()) on vérifie les 2 conditions comme suit
pour le thème on vérifie si le niveau de la question est > de dernier niveau de thème pour la question de niveau qu'on cherche on cherche si la question existe déjà dans la liste des partie déjà joué

pour le question leur numéro est composé ThemeNiveauNumqst
SAN 32 c la question 2 de thème santé de niveau 3

la méthode dernierNivIncr(theme) pour incrémenter le niveau de thème dans le cas où le joueur a trouvé la bonne réponse

on a

HashMap<Integer,Joueur> ensembleJ : qui contient la liste des joueur
ArrayList<Question> listeQ: contient la liste des question
HashSet<ThemeJeu> listeTheme: contient la liste des thème avec ses question de type LinkedList

on a ajouté une class abstait dans la classe Joueur pour donner le nombre des essaies pour chaque partie pour l'Adulte 2 et Enfant 3

on a utilisé la classe ReadWrite pour simplifier les choses dans la classe Eureka même les méthode on les a décomposer en des petites méthodes