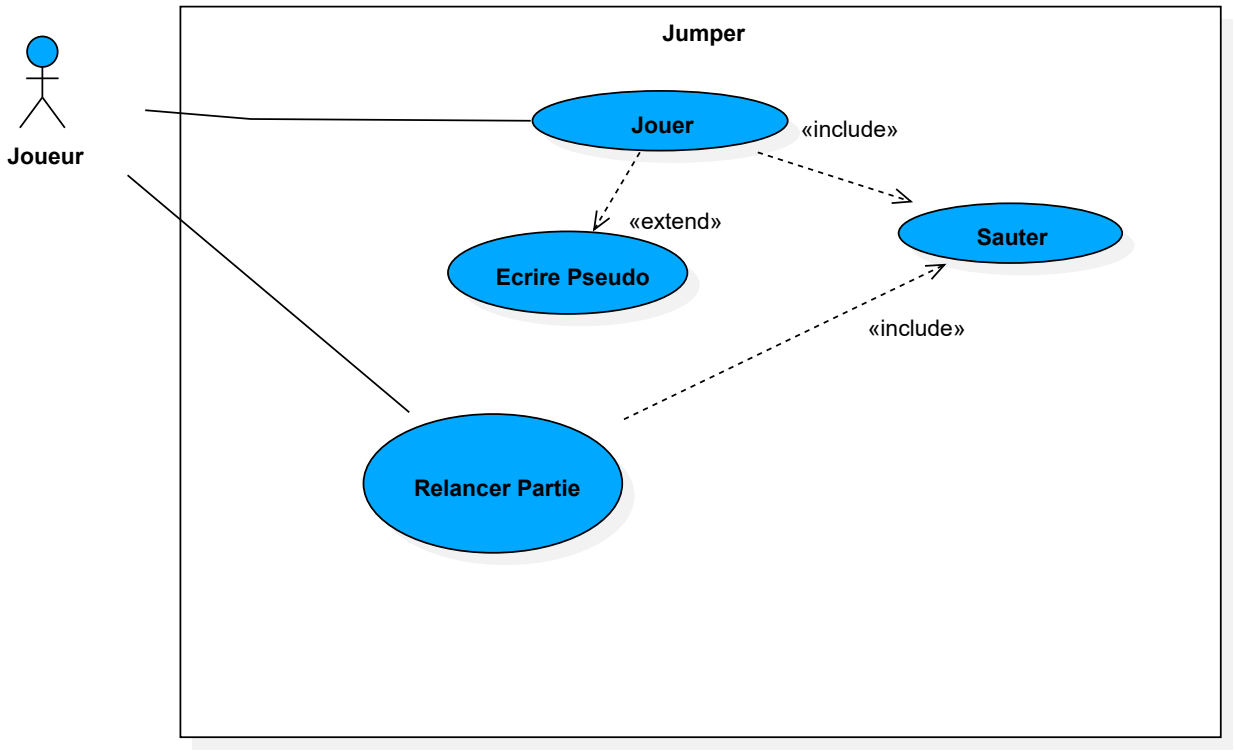
Documentation

> Ce fichier markdown contient le diagramme de classe et de cas d'utilisation du projet

CONTEXTE

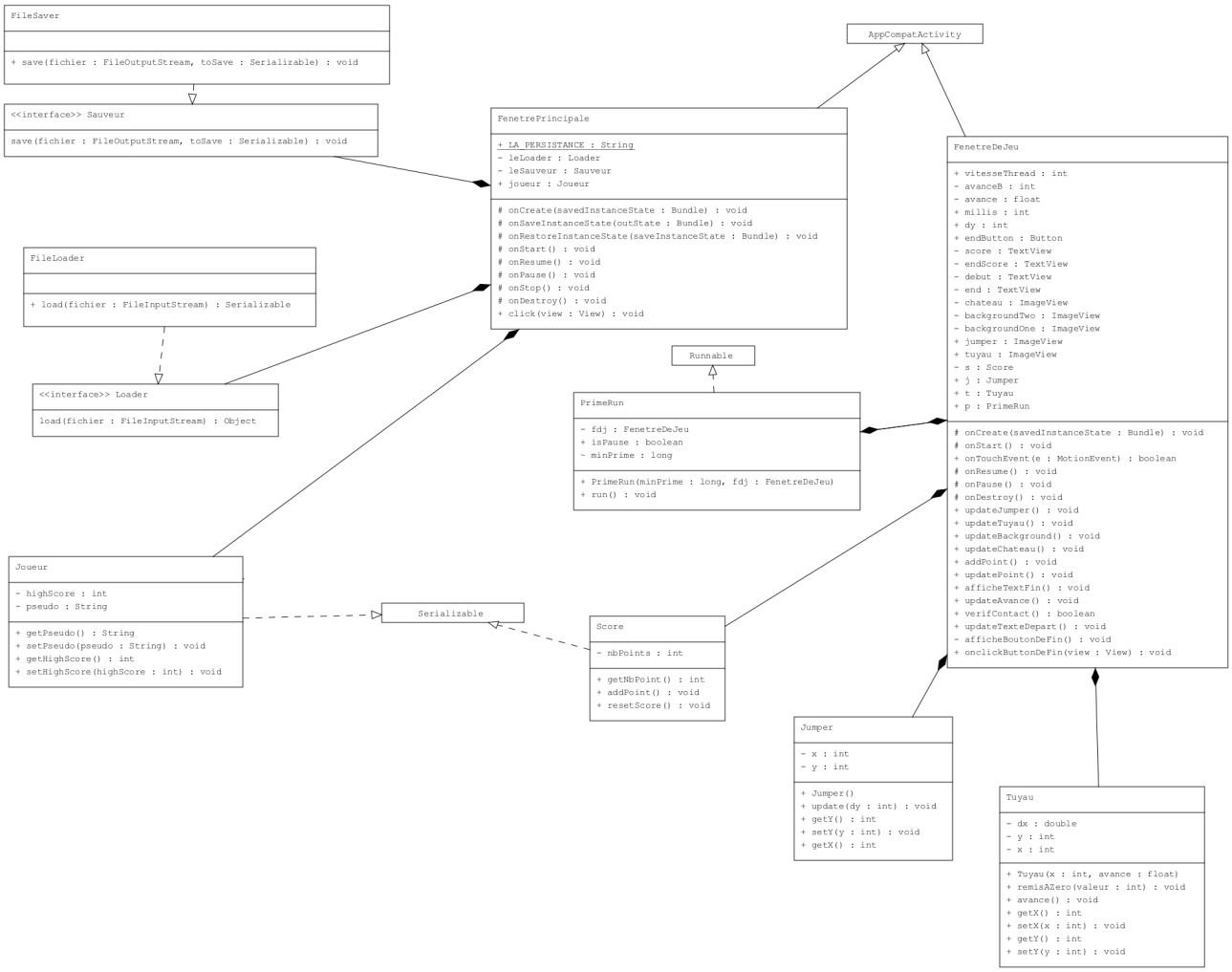
L'application Jumper est un jeu 2d développé pour la plateforme android. Le jeu est assez simple, il suffit de rentrer un pseudo et de cliquer sur commencez. Le but est d'obtenir le score le plus élevé en sautant au-dessus des obstacles en tapotant votre écran pour faire sauter Jumper. Si le personnage touche un obstacle alors la partie se termine, votre score s'affiche à l'écran : record à battre !

DIAGRAMME DE CAS D'UTILISATION

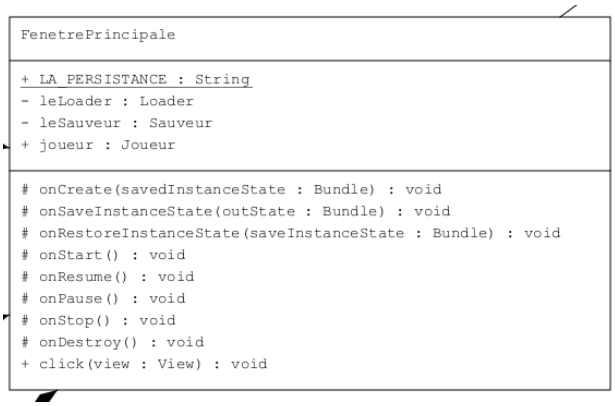


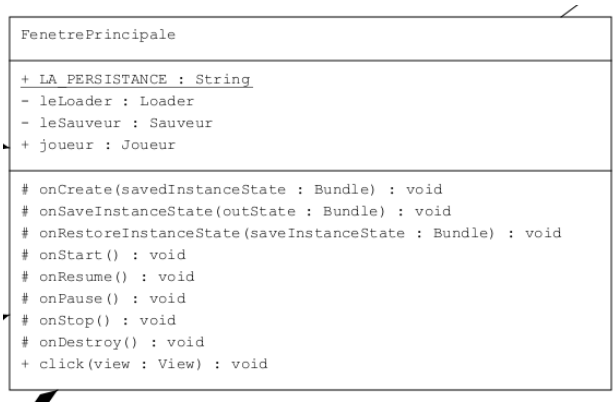
L'acteur principal de notre application est le Joueur. En premier lieu, un Joueur doit entrer son pseudo avant de pouvoir jouer. Une fois le pseudo renseigné, la Partie se lance et cela inclut le fait qu'un Joueur peut faire sauter Jumper en tapotant son écran. La partie est une boucle de jeu, tant que jumper n'a pas touché un tuyau, la partie continue en boucle tout en accélérant au fur et à mesure que les points augmentent. Une fois la partie finis, un Joueur peut relancer une partie en appuyant sur le bouton Recommencez. Un Joueur peut relancer autant de partie qu'il le souhaite pour battre son score.

DIAGRAMME DE CLASSE



FenetrePrincipale	FenetreDeJeu

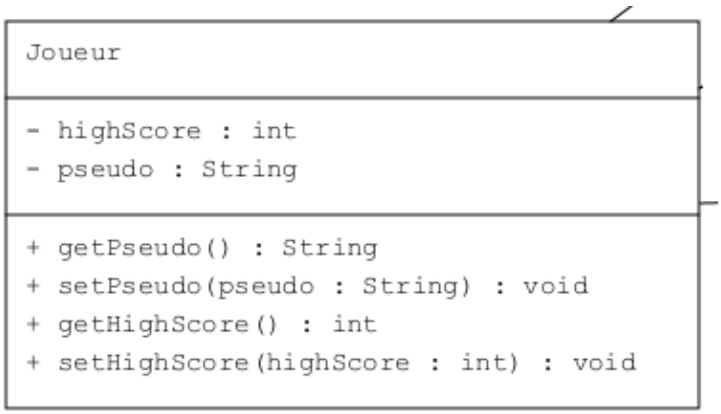
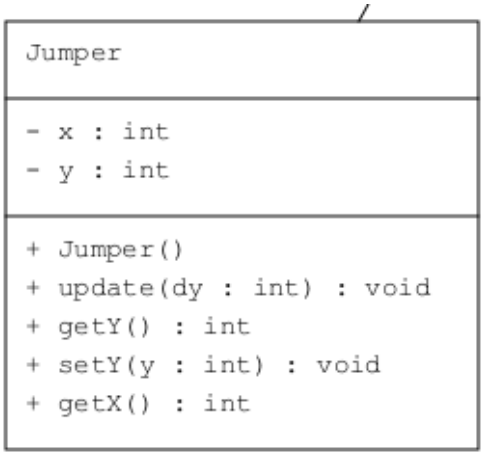




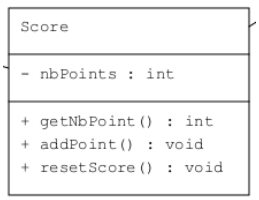
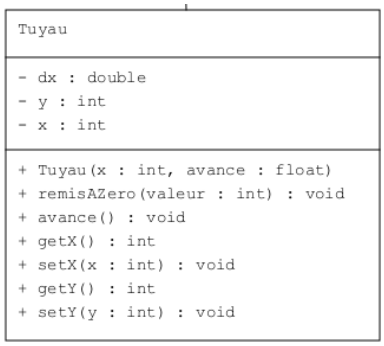
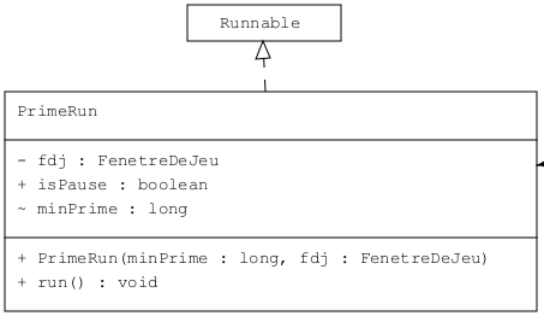
- La classe centrale de notre application est Fenetre Principale. Cette classe est rélié à 2 interface Sauveur et Loader qui permette la sauvergarde et la récupération du pseudo utilisateur. Les principales méthodes de cette classe

hérite de AppCompactActivity. La méthode click est la méthode qui permet de lancer la fenetre de jeu.

- La classe FenetreDeJeu a plusieurs attributs qui permettent de mettre en mouvement les élements de la fenêtre. La aussi les principales méthode hérite de AppCompactactivity. Par exemple onPause va automatiquement mettre le thread en pause lorsque de joueur quitte momentanément l'application. Au contraire, onResume va redémarrer le thread. Les méthodes nommés update sont les méthodes qui permettent de mettre en mouvement le jeu. La méthode afficheTextFin() permet d'afficher le bouton Recommencez du jeu. Dès que l'utilisateur clique sur le bouton Recommencez, la méthode onClickButtonDeFin est appelé. Cette méthode va redémarrer le thread et cacher à nouveau le bouton recommencer. La gestion des collisions se fait avec la méthode verifContact. A partir des coordonnées du tuyau et du Jumper, cette méthode renvoie true si jamais il y a contact.

Joueur	Jumper
	

- La classe Joueur représente un utilisateur ayant entré un pseudo. Il est représenté par un pseudo et un highScore. Ce highScore représente son meilleur score réalisé sur plusieurs parties.
- La classe Jumper a 2 attributs clés : x et y. On peut faire sauter jumper avec la méthode update qui incrémente son y.

Score	Tuyau	PrimeRun
		

- Un score est initialisé à chaque lancement de partie. Les méthodes get et add permettent de récupérer le nombre de points ou d'ajouter un point. La méthode resetScore est appelé seulement quand le joueur décide de relancer une partie
- Un tuyau est représente la aussi par un x et un y mais également un attribut dist. Cet attribut va gérer l'avance du tuyau. Cette valeur est par défaut initialisé à 5. On a la possibilité de remettre les coordonnés du tuyau a 0 avec remisAZero() mais également faire avance le tuyau, c'est à dire qu'on décrémente le y de tuyau avec la dist de 5 qu'on lui a donné.
- PrimeRun est la classe qui représente la boucle centrale du jeu. Cette classe implémente Runnable. Elle possède un attribut fdj qui représente notre fenêtre de jeu actuel mais également un boolean isPause. Tant que isPause est a false, le thread ne s'arrête pas. Hors lorsque l'utilisateur quitte la page (onResume) ou qu'il y a collision, isPause passe a true.