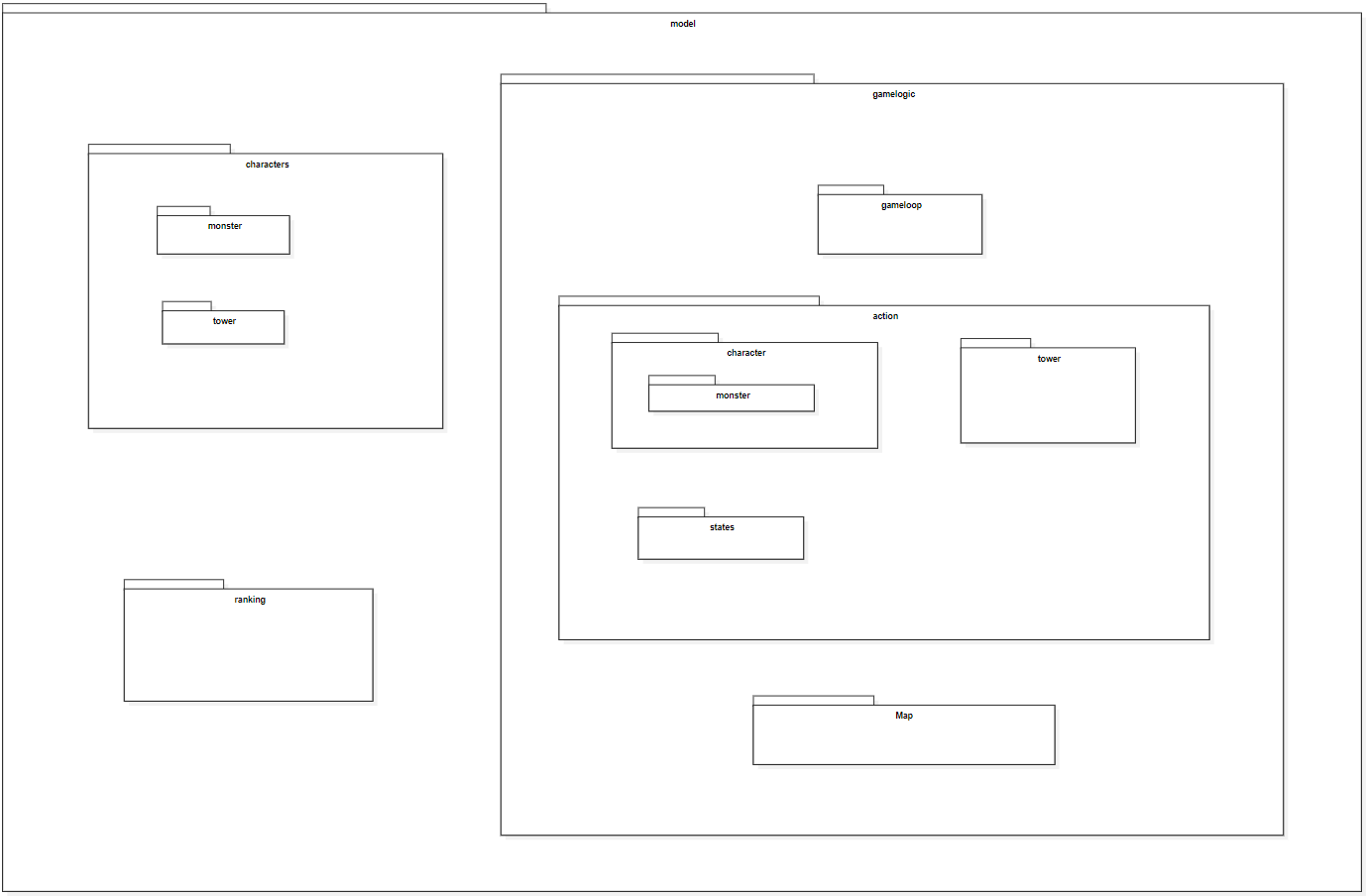
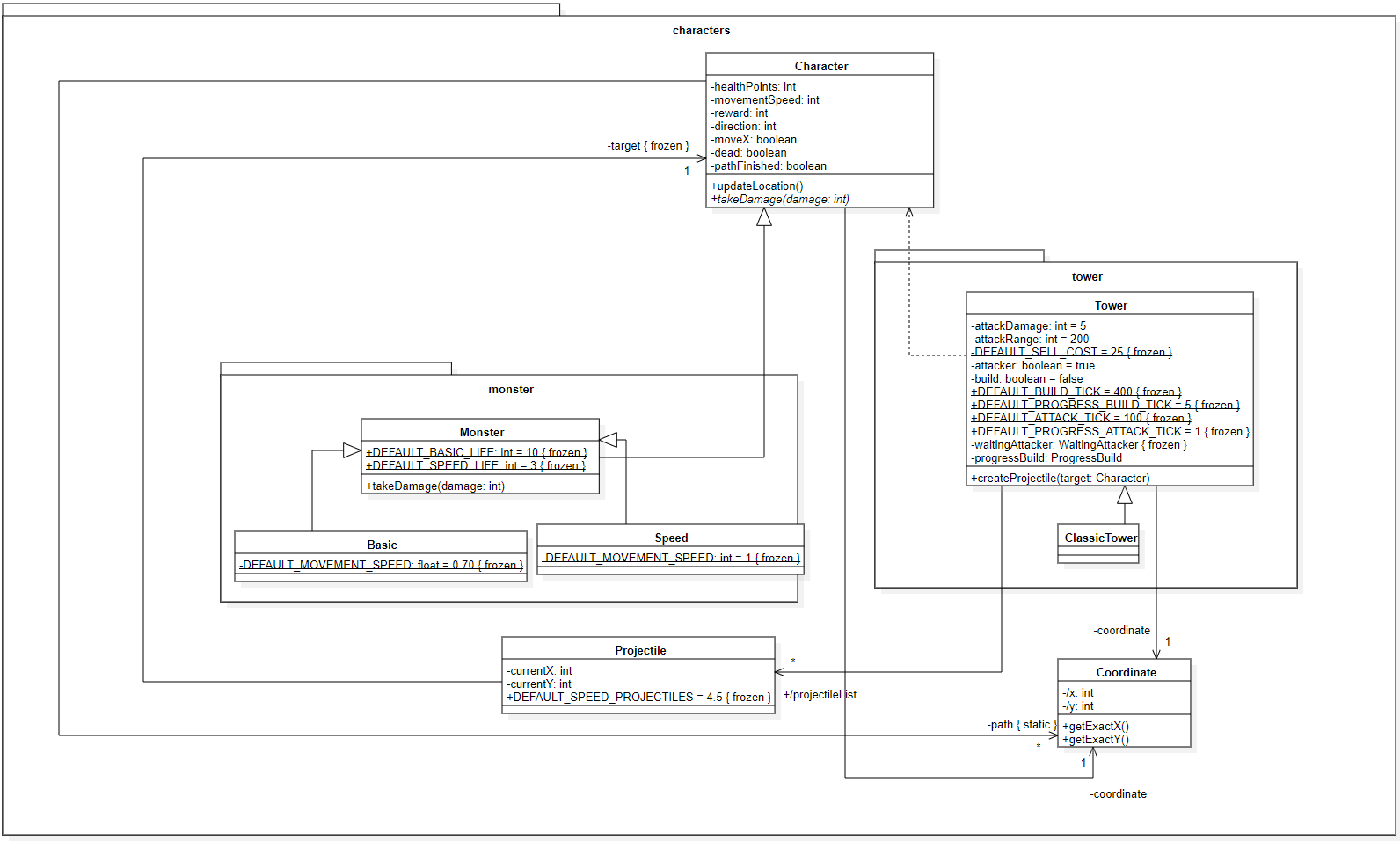
**EXPLICATIONS DIAGRAMMES DE CLASSES**

1. **Package model**



*Structuration des packages pour le model*

1.1 Package characters

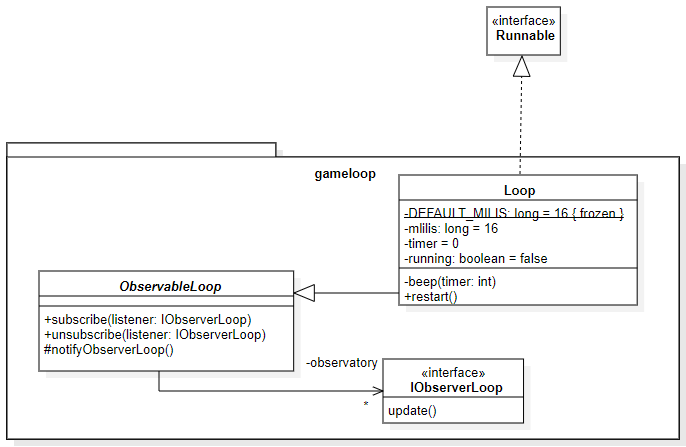
La classe Character est un point d’extensibilité et représente les entités de notre jeu. Il existe deux types de monstres les *Basic* ayant beaucoup de vie avec une vitesse faible et les *Speed* ayant une grande vitesse avec une faible vie.

La classe Coordinate représente la position d’un Character. Les méthodes *getExactX* et *getExactY* renvoient la position selon la taille des tuiles graphiques utilisées pour dessiner notre jeu.

Notre classe Tower représente une Tour de défense. Le principe est qu’elle va tirer sur les monstres (1 à la fois) s’ils sont sa zone d’attaque. Cependant avant qu’une Tour puisse attaquer il faut qu’elle se construise. La construction d’une Tour est représentée avec la classe ProgressBuild. Sa cadence d’attaque est représentée avec la classe WaitingAttacker. Ces deux classes vous seront expliquées en détails plus tard.

La classe Projectile représente les projectiles tirés par les Tours.

* 1. Package gameloop



L’idée générale est de faire tourner une boucle de jeu rythmant notre jeu. Cette boucle de jeu tourne dans un autre Thread qui n’est pas le Thread principal pour éviter que la fonction *sleep* endorme l’entièreté de notre Jeu. Pour endormir le Thread nous passons l’attribut *running* à false qui est ensuite perçu par notre boucle et l’appel à la méthode *wait* est déclenché. La méthode *restart* nous sert ensuite à réveiller notre Thread en passant *running* à true ainsi qu’en appelant la méthode *notify*. Le timer est utile pour avoir une gestion du temps et sera utilisé par le fonctionnement de notre jeu.

Notre classe Loop implémente la classe abstraite ObservableLoop, ce système nous permet de ne pas faire de l’attente active (patron de conception Observateur). Chaque tour de boucle appel la méthode *beep* qui va permettre de notifier tous les observateurs de notre boucle.