**Structure des classes:** Une classe pour chaque type personnage a été crée.

**Lunatic**: Ce constructeur possède un attribut définissant l'âge de la mort qui est déterminé de façon probabiliste en fonction d'un intervalle défini par un âge de mort minimal et maximal (fixes). L'âge minimal de reproduction est également fixé avec une probabilité de procréer. Une méthode deplacement a ensuite été ajoutée à ce constructeur de sorte à ce que l'objet lunatic puisse se déplacer de façon aléatoire dans toutes les directions (haut, bas, gauche, droite) tout en évitant d'occuper une place où un autre personnage se trouve déjà. Un switch case a été utilisé pour réaliser ce déplacement aléatoire (direction sélectionnée en fonction d'un nombre aléatoire). Une fois le déplacement fait, l'ancien emplacement du personnage est remis à vide et le nouvel emplacement est marqué d'un `L`.

**Straight**: L'âge de mort et de reproduction pour ce constructeur est déterminé de la même manière que dans le cas du constructeur Lunatic. Il est muni d'une méthode initierDeplacement qui permet de retourner un nombre aléatoire entre 0 et 3. Ce nombre sera ensuite utilisé dans une seconde méthode deplacement. Cette méthode permet au Straight de se déplacer en ligne droite en fonction du nombre aléatoire retourné par initierDeplacement. Pour ce faire, un switch case a été utilisé.

**Bad**: Le déplacement aléatoire se fait de la même manière que pour le Lunatic. La méthode deplacer permet de détecter les autres personnages en fonction de la vision du Bad (rayon de 5 par défaut). Le balayage s'opère avec une boucle for; on observe d'abord la première case depuis l'origine (emplacement actuel du Bad) en haut, droite, bas, puis gauche. Si un personnage est détecté, le Bad est forcé de se déplacer vers lui. Si aucun personnage, est détecté, le rayon augmente de 1 et ainsi de suite jusqu'à 5 (limite de la vision). Si après un balayage de 5 aucun personnage n'a été détecté, le Bad fait un pas aléatoirement et recommence le balayage visuel.

*Note: En ayant plus de leçons sur des concepts tels que l'héritage et le polymorphisme, nous aurions pu créer une classe abstraite* Personnage *qui aurait un constructeur et une méthode* Deplacement *et qui aurait pour classes filles* Lunatic, Straight, Bad. *Chacune de ces classes filles va avoir sa propre implémentation de la méthode du déplacement qui appartient à* Personnage*.*