|  |  |
| --- | --- |
|  |  |
| Rapport :Diagramme des classes |  |
|  |  |
|  |  |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | Cartes : Chaque carte appartient à une catégorie de carte (Bâtiment, Machine, Ouvrier) et c’est catégorie se différencie par leur présentation mais surtout par leurs données et leurs fonctionnalités.  Il a donc été décidé de créer une class mère abstraite « Carte », qui ne sera pas instanciable car une carte doit obligatoirement dans une catégorie, elle sera donc instanciable via les classes filles « CarteOuvrier », « CarteBatiment » et « CarteOuvrier » (classe fille de Bâtiment et donc de Carte).  Afin de gérer une pile de carte (qu’elles soient une Carte Batiment, Machine ou Ouvrier), une classe générique « PileDeCarte » a été créé. Cette classe générique n’accepte seulement que les Objets de la classe « Carte » ou des classes filles.  La class CarteFactory possède seulement 2 méthodes statiques qui permet de retourner une collection de cartes ouvriers et de bâtiments (qui comprenne les cartes machines) prédéfinie. Cela évite une récriture constante de l'instanciation des cartes. | |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | Joueur : Chaque joueur aura différent niveau de stratégique, soit les même tâches mais pas les même résultats. De ce fait, il a était décidé de créer une classe abstraite « Joueur » afin que les différentes tâches soit les même mais avec des résultats. Ces tâches sont définie dans des classes filles « BotBasique », « BotIntelligeant », « BotMedium » et « BotHard ».  La classe abstraite « BotIntelligent » permet de rajouter des methodes abstraites a ces classes filles « BotMedium » et « BotHard ». | |  |
|  |  |  |  |
|  |  |  |  |
|  | **Opération :**  Afin de différencier les différentes opérations, une classe mère abtraite à était créé « Opération ». Cela permettra d’avoir dans chaque classes filles d’avoir la methode « faireOperation() » qui permettra d’éxecuter l’opération. Ces classes filles sont « OpAffectationOuvrier », « OpAffectationBatiment », « OpObtenirEcu », « OpAffectationOuvrierBatiment » et « OpNouvelleAchat ». | |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | Partie : Afin de différencier les parties, une classe « Partie » a été créé. Chaque partie ne se finira pas de la même manière.  Afin d’exécuté et de tester les parties la classe « Main » possède la méthode « main ». Cela exécutera 1000 parties.     Statistique : Afin que le client connaisse, les détails de la partie à la fin d’une partie, la classe « Statistique » permettra d’enregistrer certaine donnée. Ces données évolueront pendant la partie et se fixeront à la fin de partie. | |  |
|  |  |  |  |