Modifications du programme principal :

- Au départ, nous avons préfixé toutes les fonctions du programme principal par le nom de la classe associée car sinon la fonction n'était pas trouvée. Par la suite, nous avons vu que l'autre façon de faire (plus propre), était d'écrire les fonctions en cascade, par exemple :
  getJoueurCourant().getPioche().piocheIsEmpty()
  au lieu de :
  Pioche.piocheIsEmpty(Joueur.getPioche(getJoueurCourant()))
  En préfixant comme nous l'avons fait, les imports ne sont pas nécessaires pour l'appel des fonctions mais ils le sont lorsque l'on crée une instance d'un objet, par exemple :
  joueur1 = Joueur()

- Les objets ne sont pas créés de la bonne façon, par exemple, il était écrit :
  newc = creerSoldat()
  au lieu de :
  newc = Carte()
  newc = creerSoldat()

- Plusieurs fonctions dans le programme principal n'étaient pas appelées correctement (inversion ou manque de paramètres, par exemple pour getCartesAttaquable)

- Quelques erreurs d'appel de fonctions comme les 2 parenthèses à la fin des appels de fonctions sans paramètre.

- Les appels de fonctions ne sont pas toujours fait avec les bons types, les objets sont parfois erronés, par exemple pour passer une carte, il y a parfois un type <str> au lieu d'un type <Carte>.
  Lorsque l'on choisit une carte de son jeu en saisissant le mot « soldat » dans la console, cette chaine de caractère n'est ensuite pas convertie en objet <Carte> dans le programme principal.

- Gestion des joueurs :
  - La création des joueurs se faisait entre la mise en place et les tours de jeu, alors qu'il fallait les créer avant la mise en place pour pouvoir s'en servir.
  - Pour simplifier les gestion des joueurs, nous avons utilisé des variables globales.
  - Parfois, ce n'était pas l'objet <Joueur> qui était manipulé mais la chaîne de caractère « joueur1 » ou « joueur2 »

- La fonction getAttaquants() qui renvoie la liste des cartes d'un joueur sur le champ de bataille doit également renvoyer la position de chacune de ces cartes car sinon il est impossible de différencier 2 cartes du même type, d'un même joueur, sur le champ de bataille. Nous avons donc retourné une seconde liste comportant les

positions.

- Il n'y a pas de gestion d'erreur :
    - erreurs de saisie
    - erreurs fonctionnelles (il est possible d'attaquer alors qu'aucune carte est attaquable, possible de démobiliser alors qu'il ne reste que le roi dans sa main)

Modifications des classes :

- Tout d'abord, nous n'avons eu besoin de rajouter aucune fonction, les classes étaient très complètes.

- Dans les fichiers des différentes classes, les fonctione de création ne prennent aucun paramètre, mais lors de la programmation, il a fallu ajouter le mot-clé « self » qui représente l'instance en cours.

- Nous avons gardé ce mot-clé « self » comme premier argument dans toutes les fonctions par convention (PEP8)

- Il manquait des imports dans les fichiers des spécifications, pour pouvoir utiliser les fonctions de certaines classes dans une autre (Carte dans Royaume, dans Main, dans Réserve, dans Cimetière, dans CDB, différents imports dans Joueur, …)

- Il n'y a pas de constructeur init pour chaque classe. Il permet d'éviter l'appel à la méthode creerCarte() et faire simplement carte = Carte() lorsque l'on souhaite créer un objet.

- La fonction ajouterRoyaume() ne devrait pas permettre d'ajouter le roi. De même pour la fonction executions(), elle doit vérifier si un roi est mort ou capturé mais il ne peut pas être au royaume.

- A l'initialisation, les cartes doivent être en position défensive et non pas offensive.

- Les 2 rois avaient les mêmes attributs, il n'y avait aucune différence.