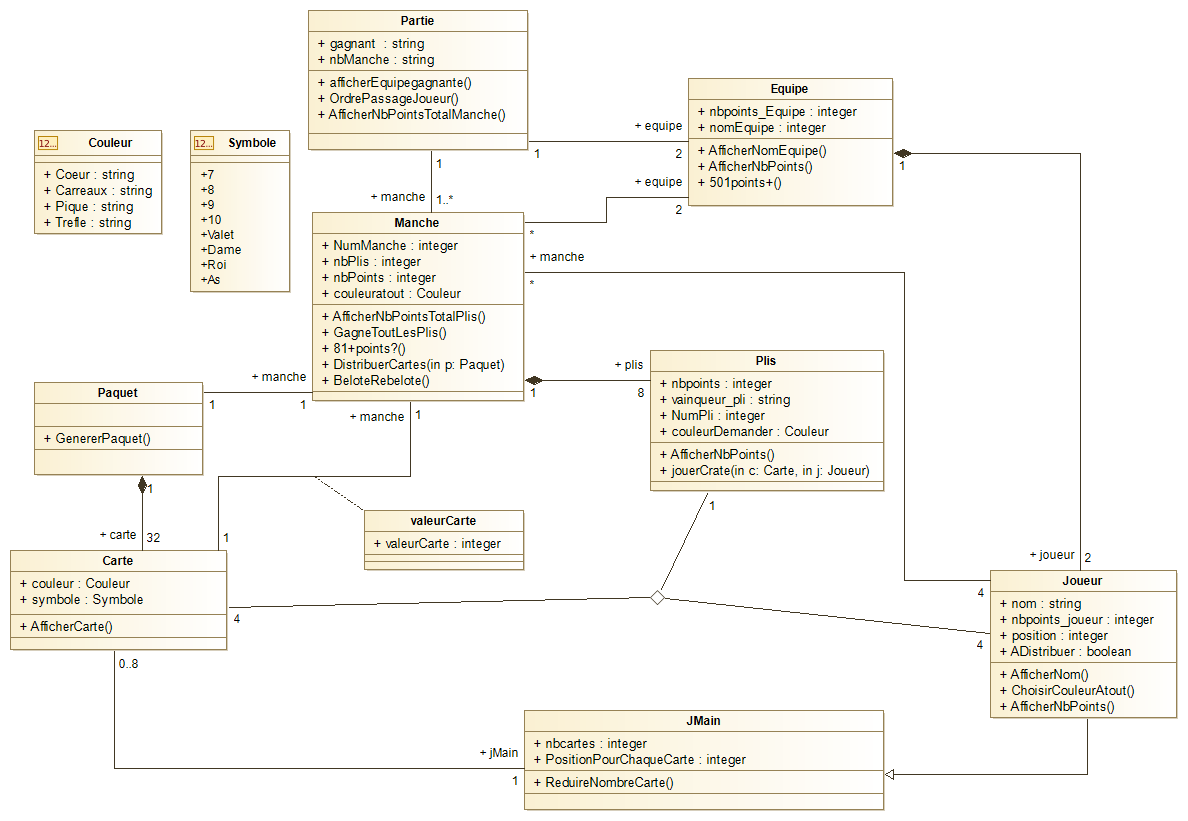
**Compte rendu DYMKO Frédéric CAZAUX Axel**

Diagramme de classe :



Contraintes ocl :

---------------------------Contrainte en jeu-------------------------------

context Plis def : cartePlusForte : Carte = self.carte -> select(c Carte | c.couleur = self.couleurDemander) -> select(c Carte | c.valeurCrate::valeurCarte -> max())

context Plis def : atoutPlusForte : Carte = self.carte -> select(c Carte | c.couleur = self.Manche.couleurAtout) -> select(c Carte | c.valeurCrate::valeurCarte -> max())

context Plis::jouerCarte(Carte c, Joueur j)

pre :

if (self.Carte -> size() = 0) then

    j.JMain.Carte -> exists(c)

else

    let mainJouable = j.JMain.Carte -> select(c Carte | c.couleur = p.couleurDemander) in

    let mainAtout = j.JMain.Carte -> select(c Carte | c.couleur = p.Manche.couleurAtout) in

    if (mainJouable -> notEmpty()) then

        if (mainJouable -> exists(c Carte | c.valeurCarte::valeurCarte > cartePlusForte.valeurCarte::valeurCarte) then

            c.valeurCarte::valeurCarte > cartePlusForte.valeurCarte::valeurCarte and

            c.couleur = self.couleurDemander and

            mainJouable -> exists(c)

        else

            if (self.couleurDemander = self.Manche.couleurAtout) then

                j.JMain.Carte -> exists(c)

            else

                c.couleur = self.couleurDemander and

                j.JMain.Carte -> exists(c)

            endif

        endif

    else

        if (self.couleurDemander = self.Manche.couleurAtout) then

            j.JMain.Carte -> exists(c)

        else

            if (mainAtout -> notEmpty()) then

                if (self.Carte -> select(c Carte | c.couleur = self.Manche.couleurAtout) -> notEmpty()) then

                    c.valeurCarte::valeurCarte > atoutPlusForte.valeurCarte::valeurCarte and

                    c.couleur = self.couleurDemander and

                    j.JMain.Carte -> exists(c)

                else

                    c.couleur = self.couleurDemander and

                    j.JMain.Carte -> exists(c)

                endif

            else

                j.JMain.Carte -> exists(c)

            endif

        endif

    endif

endif

post :

if (self.Carte -> size() = 0) then

    self.couleurDemander = c.couleur@pre and

    self.Carte = self.Carte -> append(c@pre) and

    self.Joueur = self.Joueur ->append(j@pre)

else

    self.Carte = p.Carte -> append(c@pre)

endif

------------Il n'y a pas 2x le meme joueur dans une equipe-----------------

context Equipe

inv : self.Joueur -> isUnique()

------------Un joueur n'est pas dans 2 equipes-----------------------------

context Partie

inv : self.Equipe.Joueur -> isUnique()

-----------Un joueur ne joue qu'une fois par pli---------------------------

context Plis

inv : self.Joueur -> isUnique()

-----------definition de la valeur de nbPoints dans Plis-------------------

context Plis

inv : self.nbPoints = self.Carte.valeurCarte::valeurCarte -> sum()

Dans un plis, le nombre de points est la sommes des valeurs des cartes qui ont été posé. valeurCarte ::valeurCarte est un champs contenant la valeur en points d’une carte dans une manche.

-----------Il n’y a pas de carte similaire dans le paquet------------------

Context : Carte  
Inv : Carte.allInstances() -> forAll (c1, C2| c1 <> c2 implies c1.symbole <> c2symbole or c1.couleur <> c2.couleur)

L’unicité est donc respectée par le biais de la composition entre la classe Paquet et la classe Carte qui s’occupe qu’il n’y ait que 32 cartes dans le jeu et cette contrainte OCL qui assure que pour chaque carte, le symbole ou la couleur sont différents.

Diagramme de d’activité :

