Spécif fonctionelle jeu de la crapette :

1) Carte

`couleur: Carte -> Bool`

- **Description `couleur(carte)`:** Permet d'accéder à la couleur d'une carte (ex: 1 bleu, 0 rose)

`symbole: Carte -> Int`

- **Description `symbole(carte)`:** Permet d'accéder au symbole de la carte

`numéro: Carte -> Int`

- **Description `numero(carte)`:** Permet d'accéder au numéro de la carte
- **Préconditions :**
  - `symbole` : Bool parmi les options "Pique" ou "Coeur" ou "Carreau" ou "Trèfle".

`init: symbole x Int -> Carte`

- **Description `init(symbole, numéro)`:** construit une carte de jeu

`printCarte: Carte -> String`

- **Description `printCarte(carte)`:** affiche une carte de jeu

2) Paquet

`init: String x int -> Paquet`

- **Description `init(couleur, nbCarte)`:** Crée un paquet de nbCarte cartes de la couleur spécifiée.
- **Préconditions :**
  - `couleur` : Bool parmi les options 1 ou 0.
- **Postconditions :** Renvoie une instance de Paquet contenant nbCarte Cartes de la `couleur` spécifiée.

`melange: Paquet -> Paquet`

- **Description** `melange(paquet)`**:** Mélange les cartes du paquet.
- **Préconditions :** `paquet` est une instance du type Paquet.
- **Postconditions :** Renvoie le paquet après avoir mélangé ses cartes.

`estVide: Paquet -> Bool`

- **Description** `estVide(paquet)`**:** Vérifie s'il reste des cartes dans le paquet.
- **Préconditions :** `paquet` est une instance du type Paquet.
- **Postconditions :** Renvoie un booléen : `true` si le paquet est vide, `false` sinon.

`getCarte: Paquet -> Carte`

- **Description :** donne la carte la plus haute du paquet **sans la retirer**.
- **Préconditions :** `paquet` est une instance du type Paquet.
- **Postconditions :** Renvoie la carte la plus haute du paquet.

`tirerCarte: Paquet -> Paquet`

- **Description** `tirerCarte(paquet)`**: retire** la carte la plus haute du paquet (pioche).
- **Préconditions :** `paquet` est une instance du type Paquet.
- **Postconditions :** Il y a une carte en moins dans le paquet.

3) joueur

Propriétés :

- `active` (`get` / `set`) `-> Bool` : Booléen indiquant si c'est le tour de ce joueur de jouer (`true`) ou non (`false`).
- `pseudo` (`get`) `-> String` : Chaîne de caractères représentant le nom du joueur.
- `couleur` (`get`) `-> Bool` : Chaîne de caractères indiquant la couleur attribuée au joueur (déduite de son Paquet).

`init(pseudo) -> Joueur`

- **Description :** Créé un joueur avec un pseudo et une couleur
- **Préconditions :**
  - `pseudo` : Chaîne de caractères de moins de 25 caractères.
- **Postconditions :** Renvoie une instance de Joueur

4) plateau

`init: Joueur x Joueur -> Plateau`

- **Description `init(j1, j2)`:** Initialise un plateau de jeu et l'associe aux deux joueurs. Pose les différent paquet dans les cases
- **Préconditions :**
  - `j1` et `j2` sont deux instances du type Joueur.
  - Les deux joueurs ne doivent pas avoir la même couleur.
- **Postconditions :** Renvoie une instance de Plateau initialisée. associe les emplacement 11, 12, 13 aux joueurs 1 et l'emplacement 21, 22, 23 au joueur 2. On ne sait pas qui joue encore.

`estVide: plateau x Int x Int -> Bool`

- **Description `estVide(plateau, x, y)`:** Vérifie si l'emplacement spécifié est vide.
- **Préconditions :**
  - `plateau` : L'instance du Plateau.
  - `x` : Entier compris entre 1 et 5 (inclus).
  - `y` : Entier compris entre 1 et 8 (inclus).
- **Postconditions :** Renvoie un booléen : `true` si la case est vide, `false` sinon.

`estJouable:  Plateau x Joueur x Carte x Int x Int -> Bool`

- **Description `estJouable(plateau, joueur, carte, x, y)`:** Vérifie si le placement de la carte du joueur à la position `(x, y)` est autorisé.
- **Préconditions :**
  - `plateau` : L'instance du Plateau.
  - `carte` : La carte qu'on veut poser
  - `x` : Entier compris entre 1 et 5 (inclus).
  - `y` : Entier compris entre 1 et 8 (inclus).
- **Postconditions :** Renvoie un booléen :
  - `true` si :
    1. Sur les emplacements 3 : la carte doit être d'une couleur de symbole différent et le numéro -1 de la carte sur lequel on veut poser.
    2. Sur les emplacements 4 et 5 : la carte doit être de même symbole et le numéro doit être le numéro + 1 de la carte sur lequel on veut poser.

3. Sur les emplacements adverses 1 ou 2 : La carte doit être de même symbole et le numéro +1 ou -1 de la carte sur lequel on veut poser.
    - `false` sinon.

`placer: Plateau x Joueur x Int x Int -> Plateau`

- **Description `placer(plateau, joueur, x, y)`:** Place la carte actuelle du joueur sur le plateau.
- **Préconditions :**
    - `plateau` : L'instance du Plateau.
    - `joueur` : L'instance du Joueur.
    - La fonction `estJouable(plateau, x, y)` doit être `true`.
    - Le joueur doit avoir son tour de jeu actif.
    - `x` et `y` : Entiers
- **Postconditions :**
    - Modifie le Plateau pour placer la carte.

`estCrapette: Plateau -> Bool`

- **Description `estCrapette(plateau)`:** Détermine si sur le plateau il y a une crapette.
- **Préconditions :** `plateau` est l'instance du Plateau.
- **Postconditions :**
    - Renvoie `true` si il y a une crapette sinon `false`

`estGagne: Plateau -> Joueur`

- **Description `estGagne(plateau)`:** Détermine le joueur gagnant à la fin de la manche.
- **Préconditions :** `plateau` est l'instance du Plateau.
- **Postconditions :**
    - Renvoie l'instance du type Joueur qui a gagné.
    - Le gagnant est déterminé en fonction du nombre de carte restant au joueur dans ses tas : (numJoueur, 1), (numJoueur, 2), (numJoueur, 3).
    - Renvoie `Null` s'il n'y a pas de gagnant.

`getPlateau: Plateau -> Plateau`

- **Description :** Récupère l'état actuel du jeu (du plateau). La méthode `affiche` du type Carte pourra être utilisée pour l'affichage.
- **Préconditions :** `plateau` est l'instance du Plateau.
- **Postconditions :** Renvoie l'état actuel du jeu (Plateau).

Scénario :

1) initialisation :

Les deux joueurs ont attribué 52 cartes mélangées : 4 sont placées devant eux face découvert, 12 dans leur tas (numJoueur, 3) et 36 carte dans le tas (numJoueur, 2). les deux joueurs découvrent la carte la plus haute dans leur pile de cartes (numJoueur, 3), celui dont la carte est la plus haute commence la partie.

2) déroulement :

La partie peut commencer le joueur peut transférer des cartes de paquet en paquet : si une crapette a lieu le joueur adverse peut le dire est le le joueurs arrête son coup et la main passe à l'autre joueurs. Le joueur décide si c'est la fin de son tour.

3) Fin de partie :

On regarde via la fonction **estGagné** si il y a un gagnant si oui alors le jeu s'arrête et le programme affiche le joueurs gagnant.

règle de la crapette : si le joueur pouvait poser une carte sur une case 5 ou 4 et ne la pas fait alors il est dans une situation de crapette.


51 52 53 54
23 22 21
33 32 31 34
35 36 37 38
41 42 43 44
11 12 13
11 : 0
12 : 36
13 : 12
35 : 1
36 : 1
37 : 1
41 : 0
42 : 0
43 : 0
44 : 0
21 : 0
22 : 36
23 : 12
31 : 1
32 : 1
33 : 1
51 : 0
52 : 0
53 : 0
54 : 0