Le jeu de l'équipage

Dossier de conception et code

SAÉ 1.01 – Implémentation d'un besoin client

Enzo HAMID – TD3 / TP5

Estéban DESESSARD – TD3 / TP5

Année universitaire 2023-2024

Contenu

Principe du jeu 6

Clarifications / addendum aux spécifications externes 6

Algorithme du jeu ..... 7 - 41

- Jeu de l'équipage ..... 7 - 8

-Algorithme 7

-Description des sous-problèmes 7

-Dictionnaire des variables 8

- initialiserPartie ..... 9 - 10

-Algorithme 9

-Description des sous-problèmes 9

-Dictionnaire des variables 10

- parametrerPartie ..... 11 - 12

-Algorithme 11

-Description des sous-problèmes 11

-Dictionnaire des variables 12

- jouerPartie ..... 13 - 14

-Algorithme 13

-Description des sous-problèmes 13

-Dictionnaire des variables 14

- jouerMancheJoueur ..... 15 - 16
-Algorithme 15
-Description des sous-problèmes 15
-Dictionnaire des variables 16
- jouerLancerJoueur ..... 17 - 18
-Algorithme 17
-Description des sous-problèmes 17
-Dictionnaire des variables 18
- etablirBilanLancerJoueur ..... 19 - 20
-Algorithme 19
-Description des sous-problèmes ..... 19 - 20
-Dictionnaire des variables 20
-calculerEquipementJoueur ..... 21 - 23
-Algorithme 21
-Description des sous-problèmes 22
-Dictionnaire des variables ..... 22 - 23
-verifierContinuerPartie ..... 24 - 25
-Algorithme 24
-Description des sous-problèmes 24
-Dictionnaire des variables 25

-jouerMancheMachine ..... 26 - 27
-Algorithme 26
-Description des sous-problèmes 26
-Dictionnaire des variables 27
-jouerLancerMachine ..... 28 - 29
-Algorithme 28
-Description des sous-problèmes ..... 28 - 29
-Dictionnaire des variables 29
-etablirBilanLancerMachine ..... 30 - 31
-Algorithme 30
-Description des sous-problèmes 30
-Dictionnaire des variables 31
-calculerEquipementMachine ..... 32 - 34
-Algorithme 32
-Description des sous-problèmes 33
-Dictionnaire des variables ..... 33 - 34
-verifierContinuerPartie 35
-Algorithme 35
-Description des sous-problèmes 35
-Dictionnaire des variables 35

-finaliserPartie ..... 36 - 37
-Algorithme 36
-Description des sous-problèmes 37
-Dictionnaire des variables 37
-determinerVainqueurPartie ..... 38 - 40
-Algorithme 38
-Description des sous-problèmes ..... 38 - 39
-Dictionnaire des variables 39
État de finalisation 40
Code source ..... 41 -55

Principe du jeu

En jouant à ce jeu, vous affrontez une machine sur plusieurs manches. Durant chaque manche, vous pourrez effectuer 3 lancers de 5 dés allant de 1 à 6. Selon les faces obtenues et votre équipement au cours de la manche, vous pourrez marquer ou non des points.
Dans cette version, les 5 manches se déroulent en commençant par le joueur. Le jeu peut prendre fin de deux façons différentes :

- Le joueur décide d'abandonner la partie. Le programme affiche donc le bilan de la partie avec le score du joueur ainsi que celui de la machine ;
- Le nombre total de manche est atteint et terminé. Le programme affiche alors le score final du joueur ainsi que de la machine et précise le gagnant

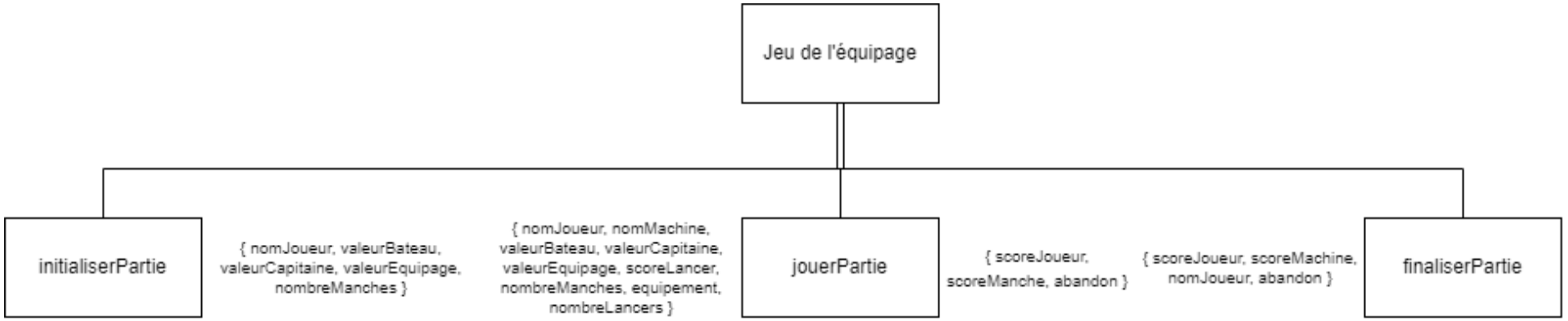
Clarifications / addendum aux spécifications externes

Aucune modification des éléments des spécifications externes

Algorithme du jeu

Jeu de l'équipage

Algorithme



Description des sous-problèmes

L'algorithme du jeu se décompose en 3 sous-problèmes principaux :
initialiserPartie : Cette étape permet d'initialiser des paramètres. Il est demandé à l'utilisateur s'il souhaite paramétrer le nombre de manches, les différents points que rapportent le bateau, le capitaine et l'équipage.
jouerPartie : Ce sous-problème correspond aux manches en alternant entre utilisateur et machine, et affichant le suivi de chaque lancer et manche à l'utilisateur.
finaliserPartie : Dans cette dernière étape, il s'agit de conclure la partie en présentant au joueur la raison pour laquelle la partie a pris fin (abandon, victoire, mot proposé inexistant, mot proposé avec des lettres interdites) ainsi que le score final.

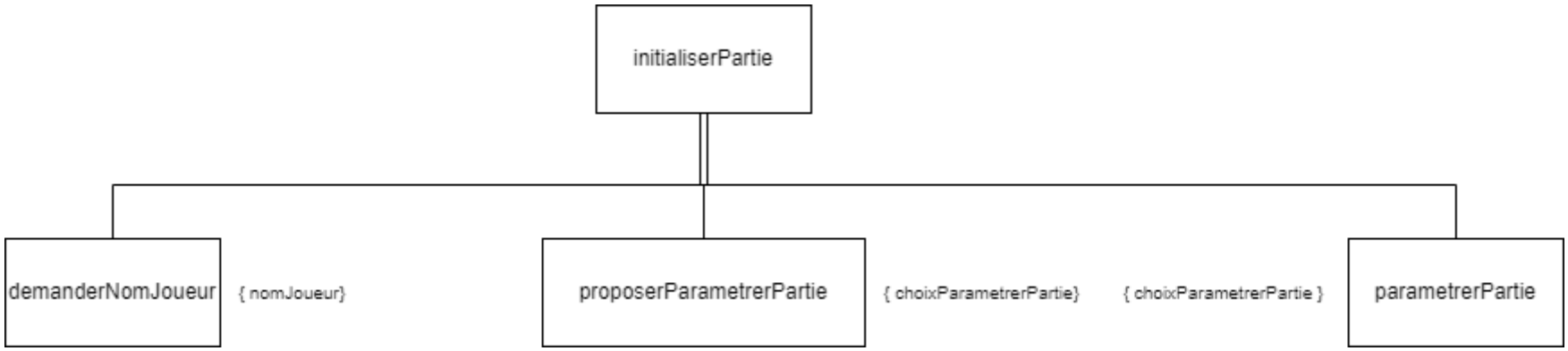
Dictionnaire des variables

Nom	Type	Signification
nomJoueur	Chaîne de caractères	Le pseudonyme que souhaite utiliser le joueur durant la partie.
valeurBateau	Nombre entier positif	Le nombre de point que rapporte l'obtention du bateau.
valeurCapitaine	Nombre entier positif	Le nombre de point que rapporte l'obtention du capitaine.
valeurEquipage	Nombre entier positif	Le nombre de point que rapporte l'obtention de l'équipage.
nombreManches	Nombre entier court positif	Le nombre de manches dans la partie

initialiserPartie

Le sous-problème **initaliserPartie** permet d'initialiser le nombre de manches et les différents points que rapportent le bateau, le capitaine et l'équipage.

Algorithme



Description des sous-problèmes

L'initialisation de la partie se décompose en trois étapes :
demanderNomJoueur : Il est demandé à l'utilisateur d'entrer le pseudonyme qu'il souhaite utiliser durant la partie.
proposerParametrerPartie : Il est demandé au joueur s'il souhaite paramétrer certains éléments de la partie.
parametrerPartie : Cette étape initialise les valeurs paramétrables à des valeurs par défaut si le joueur ne souhaite pas les modifier, ou les initialise aux valeurs que l'utilisateur entre quand elles le lui sont demandées.

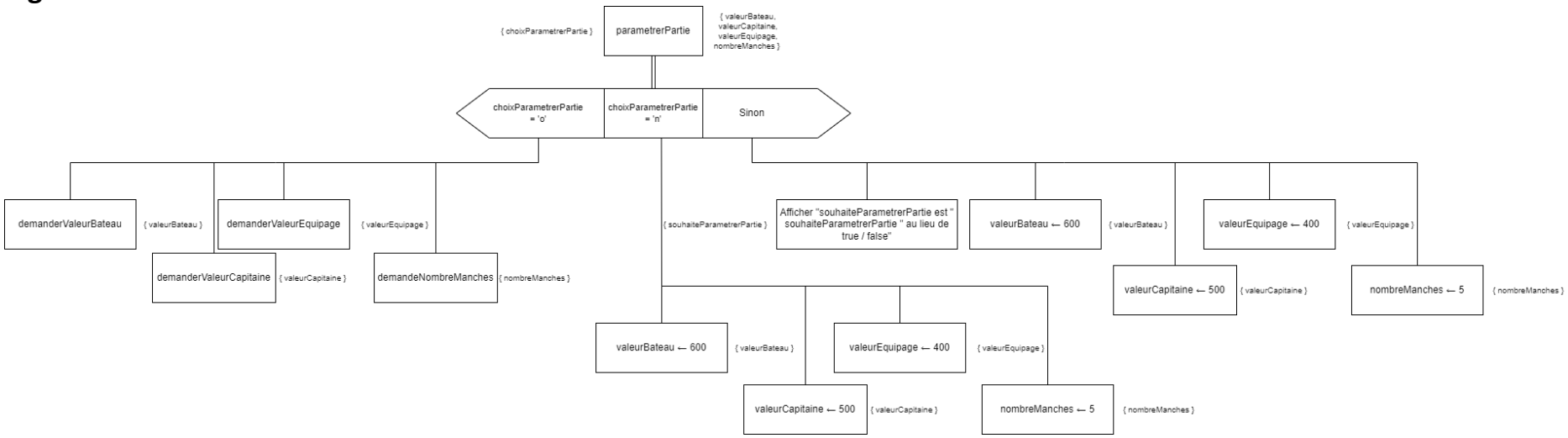
Dictionnaire des variables

Nom	**Type**	**Signification**
nomJoueur	Chaîne de caractères	Le pseudonyme que souhaite utiliser le joueur durant la partie.
choixParametrerPartie	Caractère	La réponse de l'utilisateur à la proposition de paramétrer la partie.
valeurBateau	Nombre entier positif	Le nombre de point que rapporte l'obtention du bateau.
valeurCapitaine	Nombre entier positif	Le nombre de point que rapporte l'obtention du capitaine.
valeurEquipage	Nombre entier positif	Le nombre de point que rapporte l'obtention de l'équipage.
nombreManches	Nombre entier positif	Le nombre de manches dans la partie

parametrerPartie

Le sous-problème **parametrerPartie** consiste à définir les valeurs de certains éléments de la partie.

Algorithme



Description des sous-problèmes

Le paramétrage de la partie se décompose en 3 sous-problèmes :
demanderValeurBateau : Dans cette étape, nous demandons à l'utilisateur le nombre de points que rapporte l'obtention du bateau.
demanderValeurCapitaine : Dans cette étape, nous demandons à l'utilisateur le nombre de points que rapporte l'obtention du capitaine.
demanderValeurEquipage : Dans cette étape, nous demandons à l'utilisateur le nombre de points que rapporte l'obtention de l'équipage.
demanderNombreManches : Dans cette étape, nous demandons à l'utilisateur le nombre de manches qu'il souhaite faire dans sa partie.

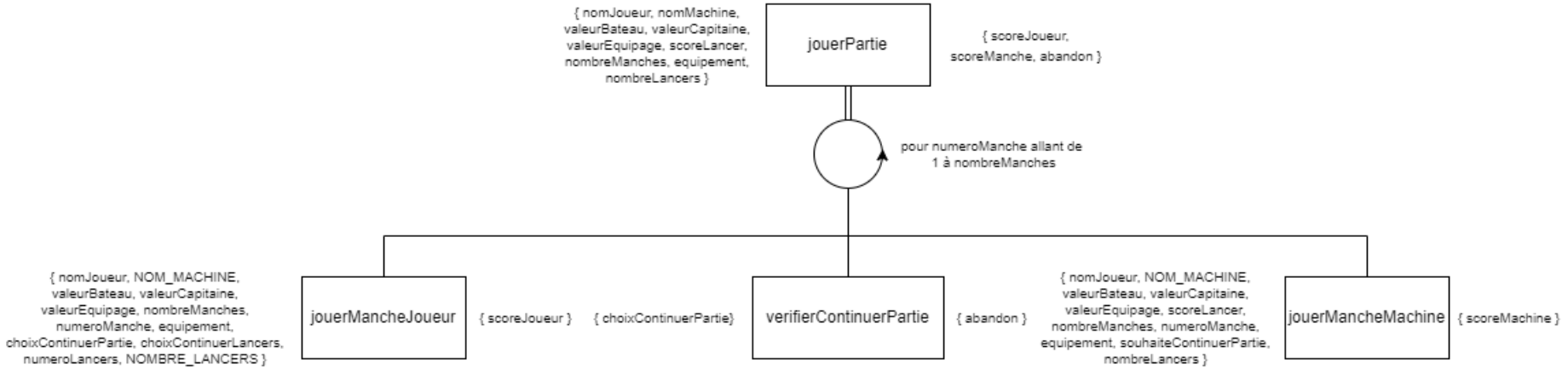
Dictionnaire des variables

Nom	**Type**	**Signification**
choixParametrerPartie	Caractère	La réponse de l'utilisateur à la proposition de paramétrer la partie.
valeurBateau	Nombre entier positif	Le nombre de point que rapporte l'obtention du bateau.
valeurCapitaine	Nombre entier positif	Le nombre de point que rapporte l'obtention du capitaine.
valeurEquipage	Nombre entier positif	Le nombre de point que rapporte l'obtention de l'équipage.
nombreManches	Nombre entier positif	Le nombre de manches dans la partie

JouerPartie

Le sous-problème **jouerPartie** gère le déroulement des différentes manches composant une partie.

Algorithme



Description des sous-problèmes

Le sous-problème **jouerPartie** consiste à répéter 3 étapes marquant le déroulement d'une manche générale. Comme indiqué dans l'algorithme, la partie peut prendre fin de 2 manières différentes : le joueur décide de ne pas continuer la partie (abandon), la boucle est terminée (toutes les manches ont été jouées).

Les 3 étapes formant le déroulement d'un tour sont les suivantes :
jouerMancheJoueur : Ce sous-problème consiste à effectuer toutes les étapes concernant la manche du joueur
verifierContinuerPartie : Cette étape consiste à vérifier si la réponse du joueur à la question de continuer la partie est positive ou négative.
jouerMancheMachine : Il s'agit ici de jouer la manche de la machine, de la même manière que celle du joueur.

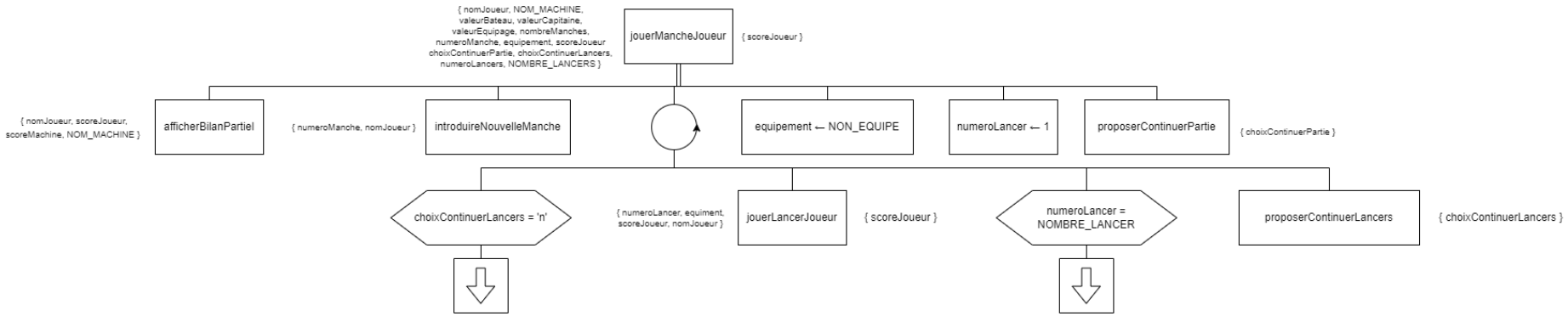
Dictionnaire des variables

Nom	Type	Signification
nomJoueur	Chaîne de caractères	Le pseudonyme que souhaite utiliser le joueur durant la partie.
NOM_MACHINE	Chaîne de caractères	Le nom de la machine.
scoreJoueur	Nombre entier positif	Le score du joueur durant la partie.
socreMachine	Nombre entier positif	Le score de la machine durant la partie.
valeurBateau	Nombre entier positif	Le nombre de point que rapporte l'obtention du bateau.
valeurCapitaine	Nombre entier positif	Le nombre de point que rapporte l'obtention du capitaine.
valeurEquipage	Nombre entier positif	Le nombre de point que rapporte l'obtention de l'équipage.
nombreManches	Nombre entier positif	Le nombre de manches dans la partie.
numeroManche	Nombre entier positif	Le numéro de la manche en cours.
equipement	Chaîne de caractères	L'équipement au cours de la manche.
choixContinuerPartie	Caractère	La réponse de l'utilisateur à la question de continuer la partie.
choixContinuerLancers	Caractère	La réponse de l'utilisateur à la question de continuer les lancers.
numeroLancer	Nombre entier court positif	Le numéro du lancer en cours.
NOMBRE_LANCERS	Nombre entier court positif	Le nombre de lancer dans une manche.

jouerMancheJoueur

Le sous-problème **jouerMancheJoueur** gère le déroulement des différentes étapes présentes dans la manche d'un joueur.

Algorithme



Description des sous-problèmes

Le sous-problème **jouerMancheJoueur** consiste à effectuer les différentes étapes présentes dans la manche du joueur

Les étapes formalisant le déroulement d'un tour sont les suivantes :
afficherBilanPartiel : Ce sous-problème consiste à afficher le bilan de la partie avant le début de la manche.
introduireNouvelleManche : Cette étape consiste à annoncer le numéro de la manche qui va se débuter.

Ensuite, le programme exécute les différents lancers de dés pour permettre au joueur de marquer des points.

Une fois que tous les lancers ont été faits, l'équipement du joueur est remis à 0 et le numéro du lancer aussi, afin de pouvoir le réutiliser pour les manches suivantes. Pour finir, nous lui proposons de continuer la partie.

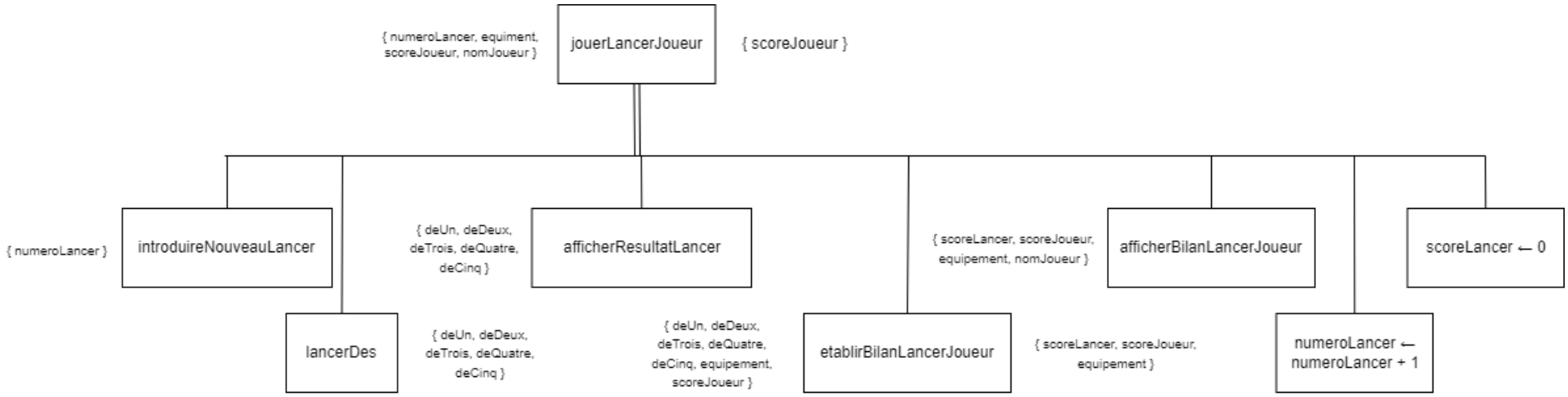
Dictionnaire des variables

Nom	Type	Signification
nomJoueur	Chaîne de caractères	Le pseudonyme que souhaite utiliser le joueur durant la partie.
NOM_MACHINE	Chaîne de caractères	Le nom de la machine.
scoreJoueur	Nombre entier positif	Le score du joueur durant la partie.
socreMachine	Nombre entier positif	Le score de la machine durant la partie.
valeurBateau	Nombre entier positif	Le nombre de point que rapporte l'obtention du bateau.
valeurCapitaine	Nombre entier positif	Le nombre de point que rapporte l'obtention du capitaine.
valeurEquipage	Nombre entier positif	Le nombre de point que rapporte l'obtention de l'équipage.
nombreManches	Nombre entier positif	Le nombre de manches dans la partie.
numeroManche	Nombre entier positif	Le numéro de la manche en cours.
equipement	Chaîne de caractères	L'équipement au cours de la manche.
choixContinuerPartie	Caractère	La réponse de l'utilisateur à la question de continuer la partie.
choixContinuerLancers	Caractère	La réponse de l'utilisateur à la question de continuer les lancers.
numeroLancer	Nombre entier court positif	Le numéro du lancer en cours.
NOMBRE_LANCERS	Nombre entier court positif	Le nombre de lancer dans une manche.

jouerLancerJoueur

Le sous-problème **jouerLancerJoueur** gère le déroulement d'un lancer de dés.

Algorithme



Description des sous-problèmes

introduireNouveauLancer : cette étape consiste à afficher le numéro du lancer qui va être effectué.
lancerDes : cette étape consiste à lancer les 5 dés à l'aide de la fonction random().
afficherResultatLancer : dans cette étape, le résultat du lancer des 5 dés est affiché.
etablirBilanLancerJoueur : cette étape consiste à calculer les modifications apportées à l'équipement du joueur, ainsi qu'à calculer son nouveau score.
afficherBilanLancerJoueur : cette étape consiste à afficher le nouvel équipement du joueur ainsi que son nouveau score.

A la fin de cette décomposition, nous incrémentons le numéro du lancer de 1.

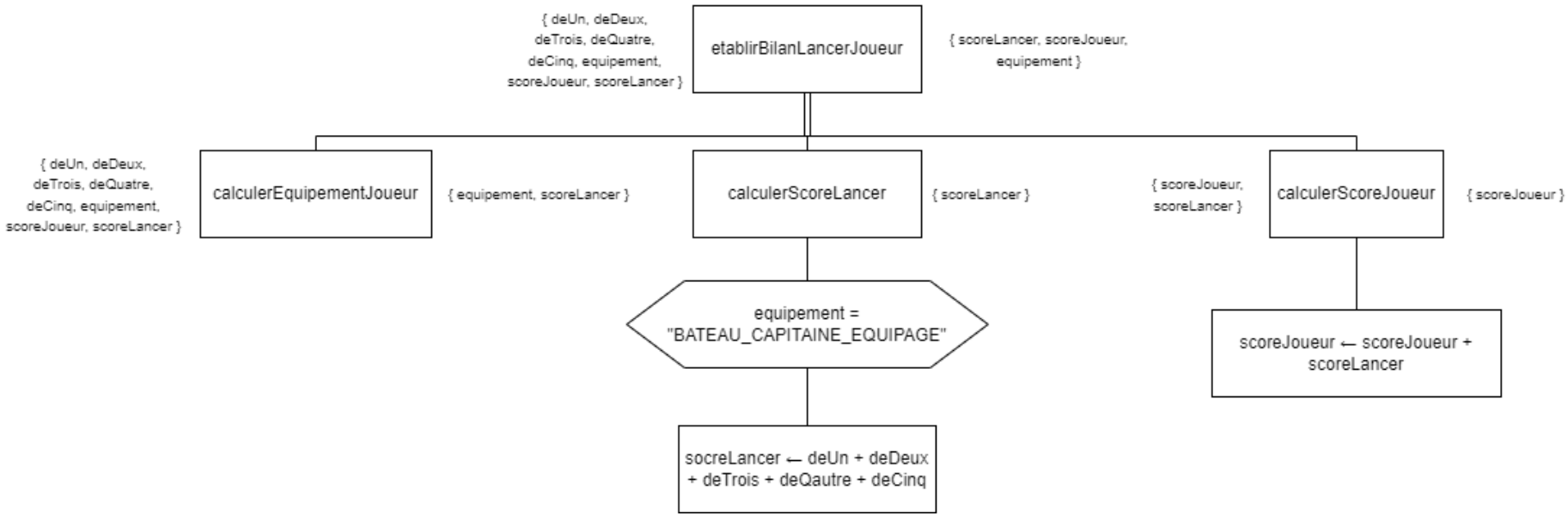
Dictionnaire des variables

Nom	Type	Signification
numeroLancer	Nombre entier court positif	Le numéro du lancer en cours.
equipement	Chaîne de caractères	L'équipement du joueur au cours de cette manche.
scoreJoueur	Nombre entier positif	Le score du joueur au long de la partie.
nomJoueur	Chaîne de caractères	Le pseudonyme utilisé par le joueur.
deUn	Nombre entier court positif	La valeur du premier dé.
deDeux	Nombre entier court positif	La valeur du deuxième dé.
deTrois	Nombre entier court positif	La valeur du troisième dé.
deQuatre	Nombre entier court positif	La valeur du quatrième dé.
deCinq	Nombre entier court positif	La valeur du cinquième dé.

etablirBilanLancerJoueur

Cette étape consiste à calculer le nouvel équipement du joueur ainsi que son nouveau score.

Algorithme



Description des sous-problèmes

calculerEquipementJoueur : cette étape consiste à calculer le nouvel équipement du joueur après le lancer des dés.
calculerScoreLancer : cette étape consiste, si la condition de l'équipement est vérifiée, à ajouter les valeurs des dés au score du lancer

calculerScoreJoueur : cette étape consiste à ajouter le score du lancer au score du joueur

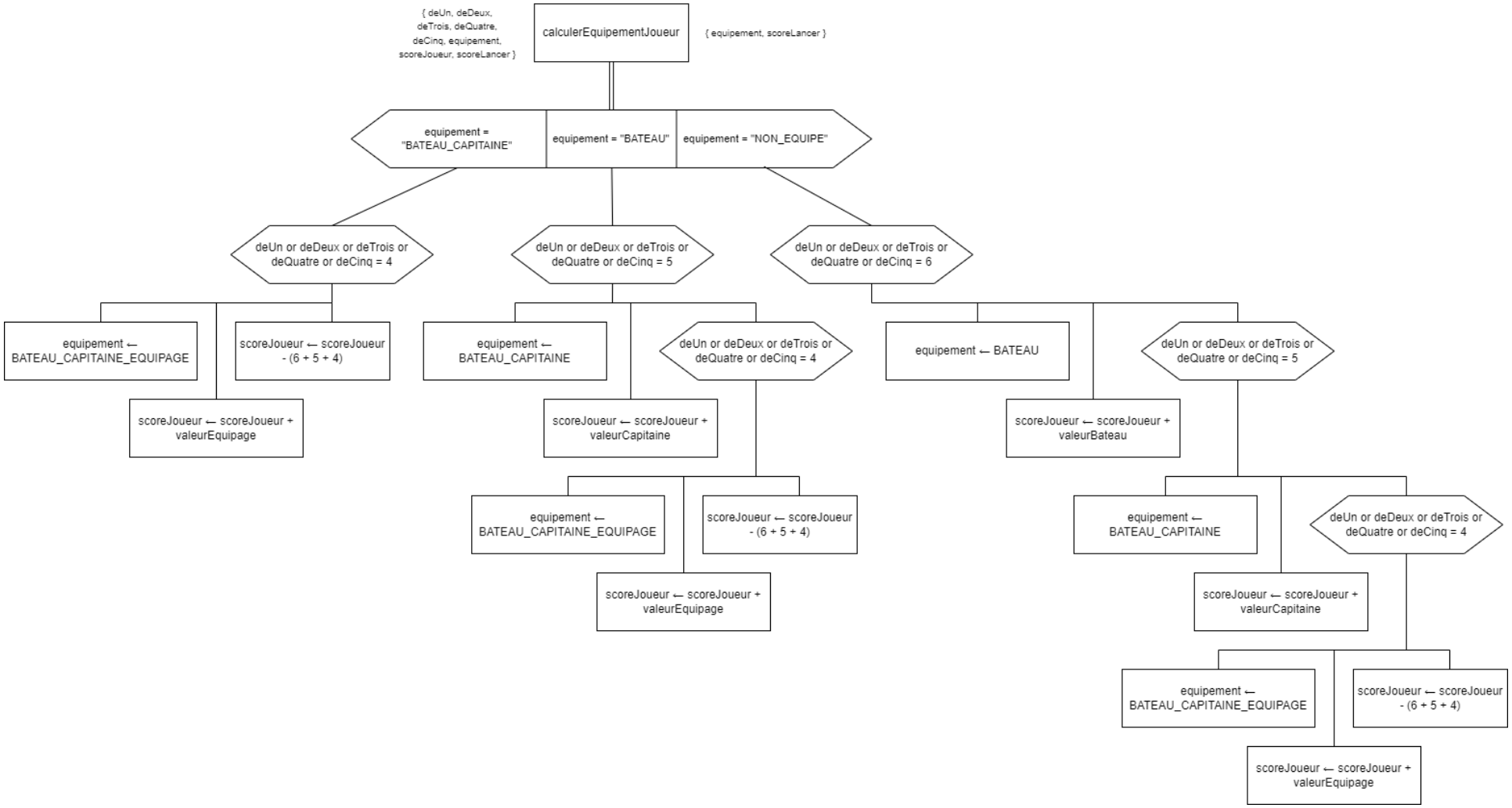
Dictionnaire des variables

Nom	**Type**	**Signification**
equipement	Chaîne de caractères	L'équipement du joueur au cours de cette manche.
scoreJoueur	Nombre entier positif	Le score du joueur au long de la partie.
deUn	Nombre entier court positif	La valeur du premier dé.
deDeux	Nombre entier court positif	La valeur du deuxième dé.
deTrois	Nombre entier court positif	La valeur du troisième dé.
deQuatre	Nombre entier court positif	La valeur du quatrième dé.
deCinq	Nombre entier court positif	La valeur du cinquième dé.

calculerEquipementJoueur

L'étape **calculerEquipementJoueur** consiste à, une fois le lancer des dés effectués, calculer le nouvel équipement du joueur selon les faces de dés obtenues.

Algorithme



Description des sous-problèmes

Ce sous-problème se décompose en une analyse de l'état général de l'équipement du joueur. Selon l'état de son équipement actuel, celui-ci sera mis à jour de différentes manières :
Si le joueur possède aucun équipement : le programme vérifie d'abord si un 6 a été obtenu. Si c'est le cas, l'équipement du joueur ainsi que le score du lancer sont mis à jour et le programme vérifie par la suite si un 5 a été obtenu. Si c'est le cas également, le capitaine est ajouté à l'équipement du joueur et le score du lancer est incrémenter du nombre de points que rapport l'obtention du bateau. Pour finir, le programme vérifiera si un 4 a été obtenu, correspondant à l'ajout de l'équipage à l'équipement du joueur ; s'en suit donc l'ajout du nombre de point rapportés par l'équipage au score du lancer. Sans oublier, le retrait de 6 + 5 + 4 points du score général du joueur, pour ne pas compter deux fois les dés permettant l'obtention de l'équipement.
Si le joueur possède déjà le bateau : si le joueur possède déjà le bateau dans on équipement, alors les mêmes étapes de vérifications sont effectuées que dans la condition précédente, en passant uniquement la première, consistant à vérifier si un 6 a été obtenu.
Si le joueur possède déjà le bateau et le capitaine : si le joueur possède déjà ces deux éléments dans son équipement, la dernière étape de vérification est effectuée, à savoir si le joueur à obtenu ou non un dé avec la face 4 sur celui-ci.
Si le joueur possède déjà tout son équipement : si le joueur possède déjà tout l'équipement nécessaire pour marquer des points, alors seul le calcul du score du lancer est effectué dans le sous-problème suivant celui-ci.

Dictionnaire des variables

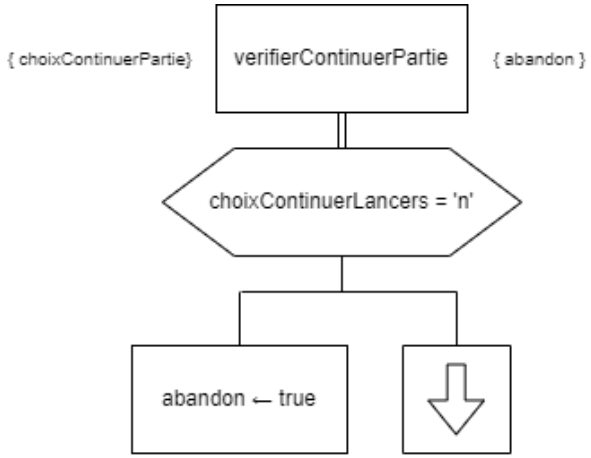
Nom	**Type**	**Signification**
equipement	Chaîne de caractères	L'équipement du joueur au cours de cette manche.
scoreJoueur	Nombre entier positif	Le score du joueur au long de la partie.
deUn	Nombre entier court positif	La valeur du premier dé.
deDeux	Nombre entier court positif	La valeur du deuxième dé.
deTrois	Nombre entier court positif	La valeur du troisième dé.

deQuatre	Nombre entier court positif	La valeur du quatrième dé.
deCinq	Nombre entier court positif	La valeur du cinquième dé.
scoreLancer	Nombre entier positif	Le score du lancer en cours.

verifierContinuerPartie

Une fois sa manche terminée, le joueur peut décider d'abandonner la partie ou non. À ce moment-là, la machine gagnera, peu importe le score de chacun. Cette étape consiste donc à vérifier la réponse du joueur à la question de continuer la partie.

Algorithme



Description des sous-problèmes

Seule une condition est vérifiée, si la réponse du joueur est 'n' (pour « Non »). Si c'est le cas, alors le booléen *abandon* est défini sur *true* pour indiquer au programme lors de la fin de partie, que celle-ci a été terminée par abandon.

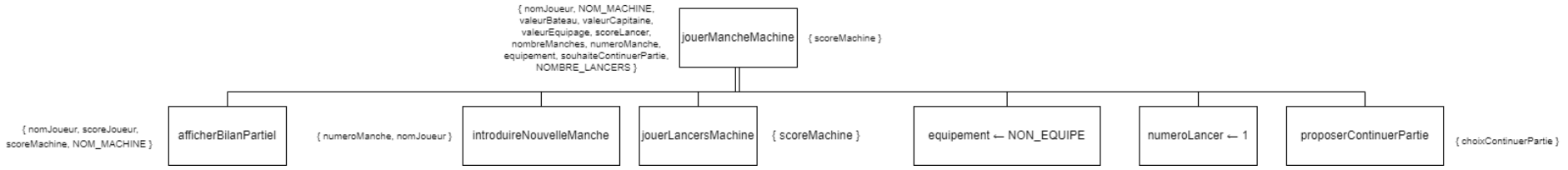
Dictionnaire des variables

Nom	**Type**	**Signification**
choixContinuerPartie	Caractère	La réponse du joueur à la question de continuer la partie.
abandon	Booléen	Défini si la partie s'est terminée par abandon ou non.

jouerMancheMachine

Tout comme le joueur, cette étape consiste à jouer la manche de la machine, comportant notamment les 3 lancers de dés.

Algorithme



Description des sous-problèmes

Le sous-problème **jouerMancheMachine** consiste à effectuer les différentes étapes présentes dans la manche du joueur

Les étapes formalisant le déroulement d'un tour sont les suivantes :
afficherBilanPartiel : Ce sous-problème consiste à afficher le bilan de la partie avant le début de la manche.
introduireNouvelleManche : Cette étape consiste à annoncer le numéro de la manche qui va se débuter.
jouerLancersMachine : Le principe de ce sous-problème est de jouer les 3 lancers de dés de la machine

Une fois que tous les lancers ont été faits, l'équipement de la machine est remis à 0 et le numéro du lancer aussi, afin de pouvoir le réutiliser pour les manches suivantes.

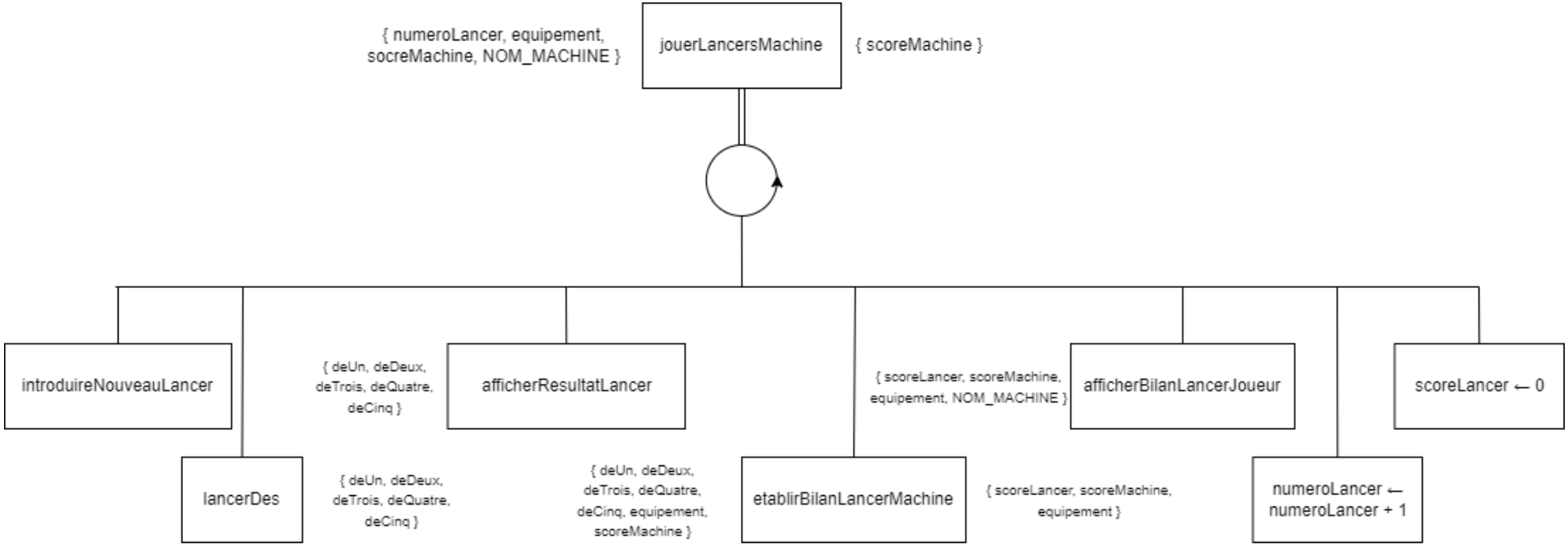
Dictionnaire des variables

Nom	Type	Signification
nomJoueur	Chaîne de caractères	Le pseudonyme que souhaite utiliser le joueur durant la partie.
NOM_MACHINE	Chaîne de caractères	Le nom de la machine.
scoreLancer	Nombre entier positif	Le score du lancer effectué.
socreMachine	Nombre entier positif	Le score de la machine durant la partie.
valeurBateau	Nombre entier positif	Le nombre de point que rapporte l'obtention du bateau.
valeurCapitaine	Nombre entier positif	Le nombre de point que rapporte l'obtention du capitaine.
valeurEquipage	Nombre entier positif	Le nombre de point que rapporte l'obtention de l'équipage.
nombreManches	Nombre entier positif	Le nombre de manches dans la partie.
numeroManche	Nombre entier positif	Le numéro de la manche en cours.
equipement	Chaîne de caractères	L'équipement au cours de la manche.
choixContinuerLancers	Caractère	La réponse de l'utilisateur à la question de continuer les lancers.
numeroLancer	Nombre entier court positif	Le numéro du lancer en cours.
NOMBRE_LANCERS	Nombre entier court positif	Le nombre de lancer dans une manche.

JouerLancerMachine

Tout comme du côté du joueur, cette étape consiste à effectuer les lancers de dés pour permettre à la machine de marquer des points. Seul point différent majeur, nous ne lui proposons pas de continuer les lancers. Nous proposons uniquement au joueur de continuer la partie à la fin de la manche de la machine.

Algorithme



Description des sous-problèmes

introduireNouveauLancer : cette étape consiste à afficher le numéro du lancer qui va être effectué.

lancerDes : cette étape consiste à lancer les 5 dés à l'aide de la fonction random().
afficherResultatLancer : dans cette étape, le résultat du lancer des 5 dés est affiché.
etablirBilanLancerMachine : cette étape consiste à calculer les modifications apportées à l'équipement du de la machine, ainsi qu'à calculer son nouveau score.
afficherBilanLancerMachine : cette étape consiste à afficher le nouvel équipement de la machine ainsi que son nouveau score.
A la fin de cette décomposition, nous incrémentons le numéro du lancer de 1.

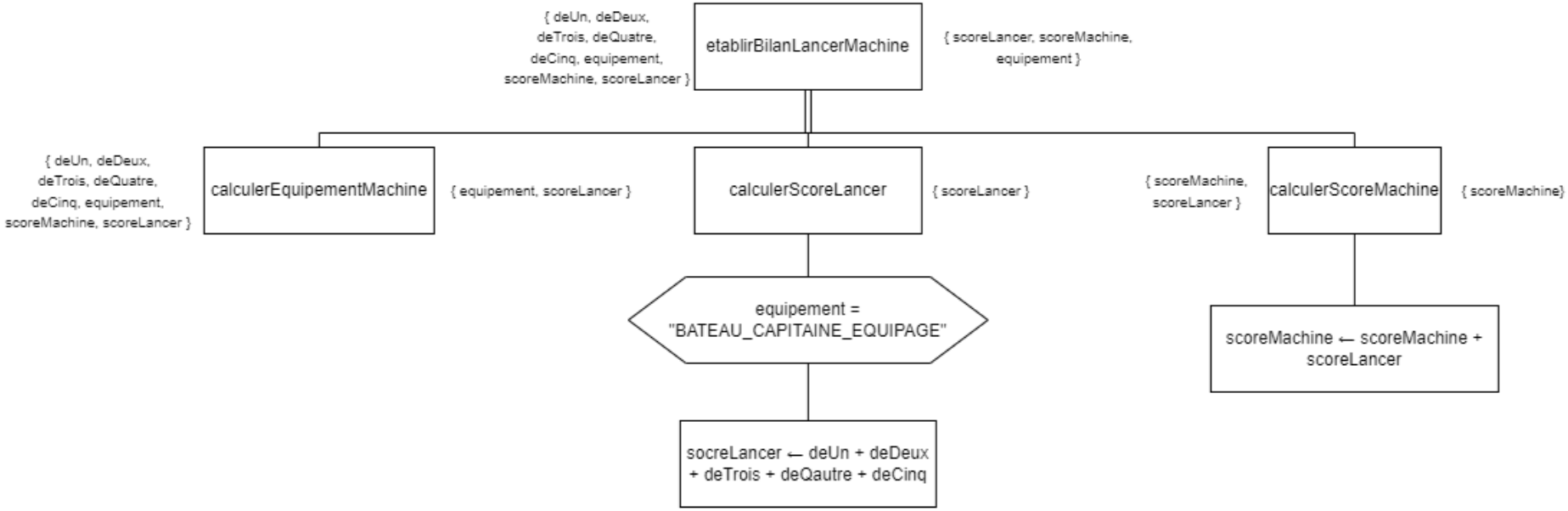
Dictionnaire des variables

Nom	**Type**	**Signification**
numeroLancer	Nombre entier court positif	Le numéro du lancer en cours.
equipement	Chaîne de caractères	L'équipement de la machine au cours de cette manche.
scoreMachine	Nombre entier positif	Le score de la machine au long de la partie.
NOM_MACHINE	Chaîne de caractères	Le nom de la machine
deUn	Nombre entier court positif	La valeur du premier dé.
deDeux	Nombre entier court positif	La valeur du deuxième dé.
deTrois	Nombre entier court positif	La valeur du troisième dé.
deQuatre	Nombre entier court positif	La valeur du quatrième dé.
deCinq	Nombre entier court positif	La valeur du cinquième dé.

etablirBilanLancerMachine

Cette étape consiste à calculer le nouvel équipement de la machine ainsi que son nouveau score.

Algorithme



Description des sous-problèmes

calculerEquipementMachine : cette étape consiste à calculer le nouvel équipement du joueur après le lancer des dés.
calculerScoreLancer : cette étape consiste, si la condition de l'équipement est vérifiée, à ajouter les valeurs des dés au score du lancer
calculerScoreMachine : cette étape consiste à ajouter le score du lancer au score du joueur

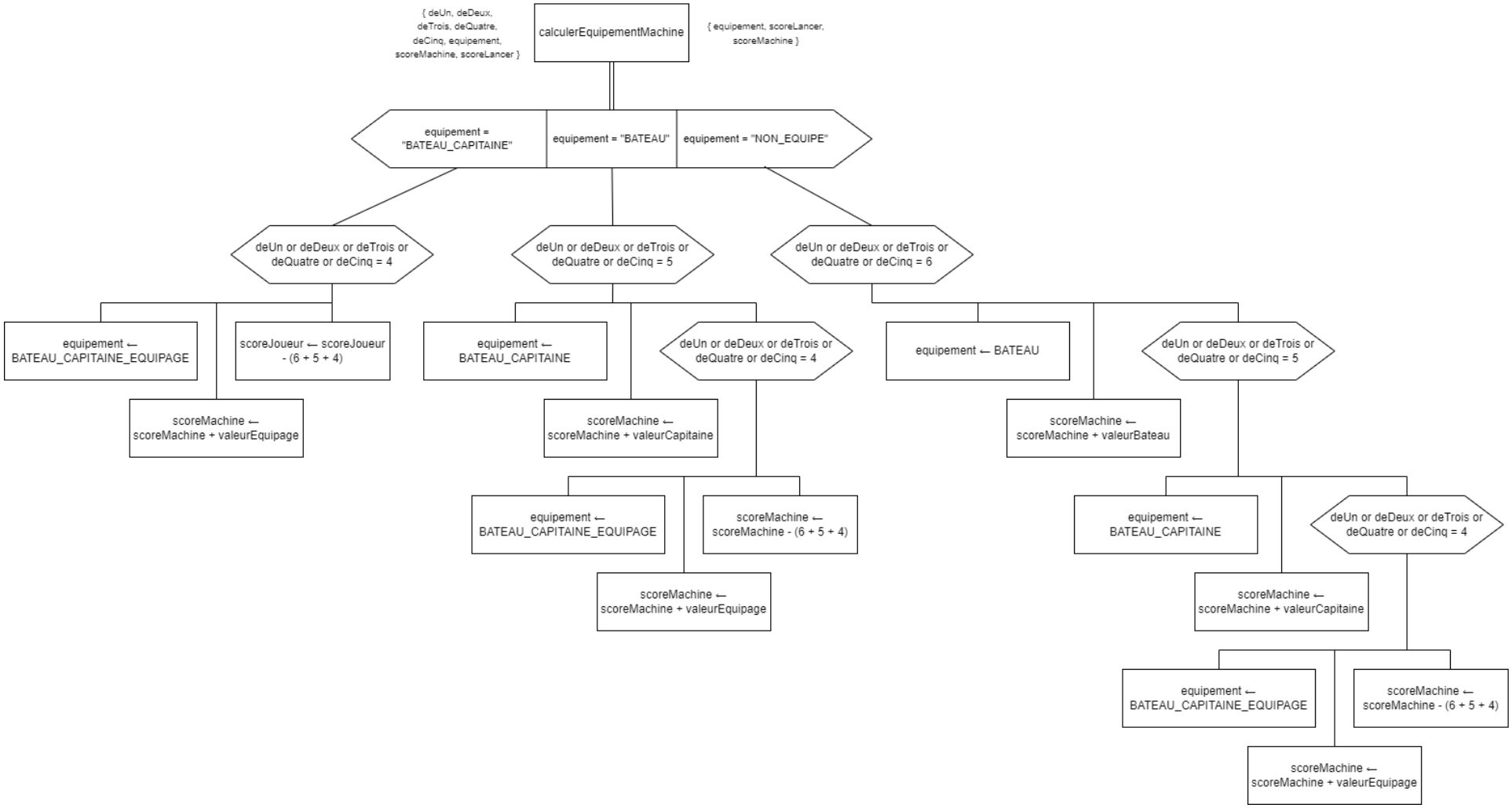
Dictionnaire des variables

Nom	**Type**	**Signification**
equipement	Chaîne de caractères	L'équipement de la machine au cours de cette manche.
scoreMachine	Nombre entier positif	Le score de la machine au long de la partie.
deUn	Nombre entier court positif	La valeur du premier dé.
deDeux	Nombre entier court positif	La valeur du deuxième dé.
deTrois	Nombre entier court positif	La valeur du troisième dé.
deQuatre	Nombre entier court positif	La valeur du quatrième dé.
deCinq	Nombre entier court positif	La valeur du cinquième dé.

calculerEquipementMachine

L'étape **calculerEquipementJoueur** consiste à, une fois le lancer des dés effectués, calculer le nouvel équipement du joueur selon les faces de dés obtenues.

Algorithme



Description des sous-problèmes

Ce sous-problème se décompose en une analyse de l'état général de l'équipement de la machine. Selon l'état de son équipement actuel, celui-ci sera mis à jour de différentes manières :
Si la machine possède aucun équipement : le programme vérifie d'abord si un 6 a été obtenu. Si c'est le cas, l'équipement de la machine ainsi que le score du lancer sont mis à jour et le programme vérifie par la suite si un 5 a été obtenu. Si c'est le cas également, le capitaine est ajouté à l'équipement de la machine et le score du lancer est incrémenter du nombre de points que rapport l'obtention du bateau. Pour finir, le programme vérifiera si un 4 a été obtenu, correspondant à l'ajout de l'équipage à l'équipement de la machine ; s'en suit donc l'ajout du nombre de point rapportés par l'équipage au score du lancer. Sans oublier, le retrait de 6 + 5 + 4 points du score général de la machine, pour ne pas compter deux fois les dés permettant l'obtention de l'équipement.
Si la machine possède déjà le bateau : si la machine possède déjà le bateau dans on équipement, alors les mêmes étapes de vérifications sont effectuées que dans la condition précédente, en passant uniquement la première, consistant à vérifier si un 6 a été obtenu.
Si la machine possède déjà le bateau et le capitaine : si la machine possède déjà ces deux éléments dans son équipement, la dernière étape de vérification est effectuée, à savoir si la machine a obtenu ou non un dé avec la face 4 sur celui-ci.
Si la machine possède déjà tout son équipement : si la machine possède déjà tout l'équipement nécessaire pour marquer des points, alors seul le calcul du score du lancer est effectué dans le sous-problème suivant celui-ci.

Dictionnaire des variables

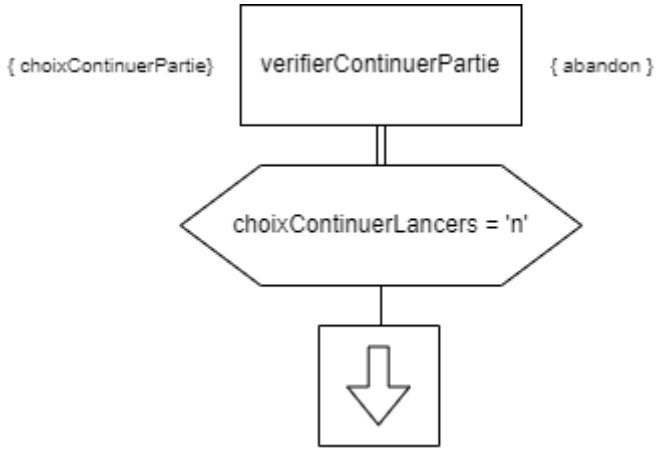
Nom	**Type**	**Signification**
equipement	Chaîne de caractères	L'équipement du joueur au cours de cette manche.
scoreMachine	Nombre entier positif	Le score de la machine au long de la partie.
deUn	Nombre entier court positif	La valeur du premier dé.
deDeux	Nombre entier court positif	La valeur du deuxième dé.
deTrois	Nombre entier court positif	La valeur du troisième dé.

deQuatre	Nombre entier court positif	La valeur du quatrième dé.
deCinq	Nombre entier court positif	La valeur du cinquième dé.
scoreLancer	Nombre entier positif	Le score du lancer en cours.

verifierContinuerPartie

Une fois la manche de la machine terminée, le joueur peut décider d'abandonner la partie ou non. À ce moment-là, la machine gagnera, peu importe le score de chacun. Cette étape consiste donc à vérifier la réponse du joueur à la question de continuer la partie.

Algorithme



Description des sous-problèmes

Seule une condition est vérifiée, si la réponse du joueur est 'n' (pour « Non »), si oui la boucle se termine.

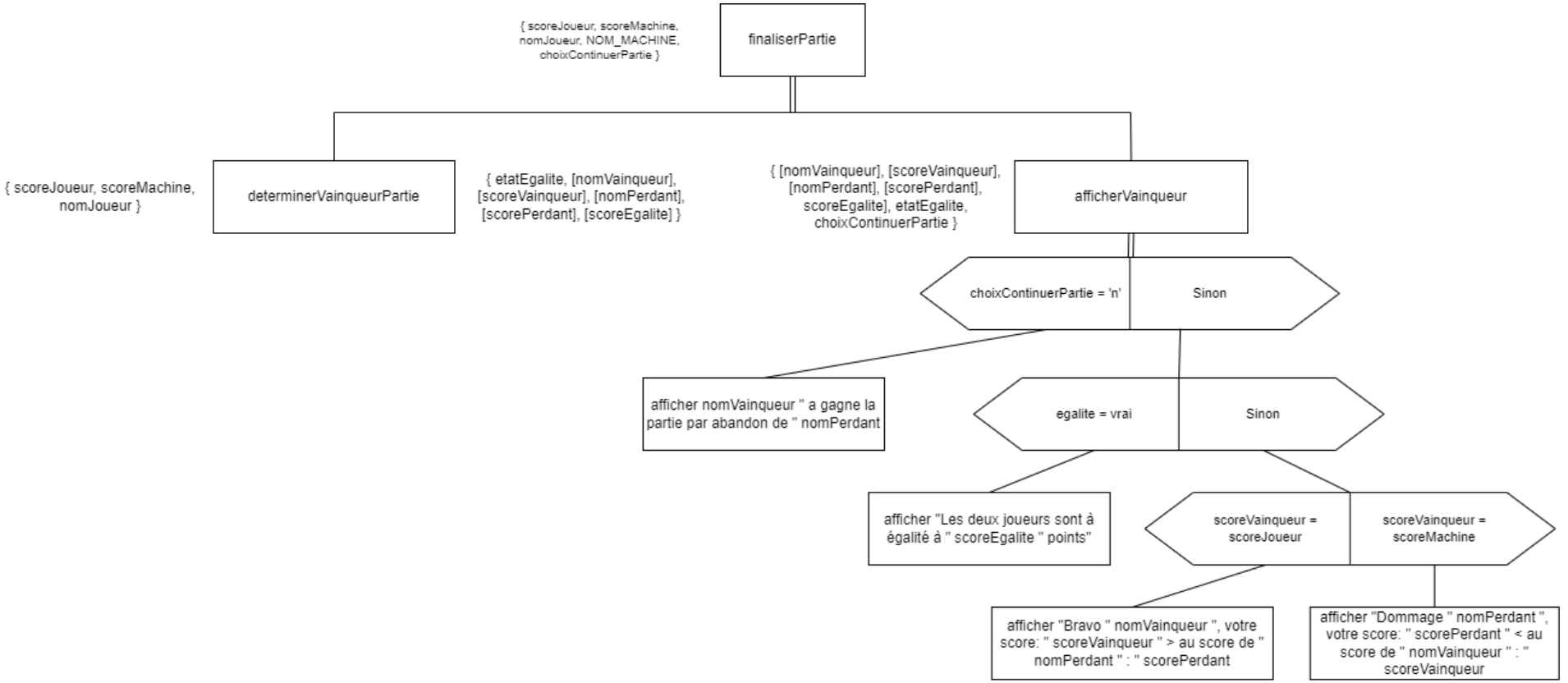
Dictionnaire des variables

Nom	**Type**	**Signification**
choixContinuerPartie	Caractère	La réponse du joueur à la question de continuer la partie.

finaliserPartie

Une fois toutes les manches effectuées, ou bien par abandon du joueur, nous arrivons au dernier sous-problème de la décomposition d'une partie : la finalisation. Cette étape est divisée en deux autres problèmes majeurs.

Algorithme



Description des sous-problèmes

determinerVainqueurPartie : cette étape consiste avant tout à vérifier si le joueur à abandonner ou non. Ensuite, nous comparons le score de la machine avec celui du joueur et déterminons le vainqueur.
afficherVainqueurPartie : cette étape consiste à afficher le vainqueur de la partie avec son score.

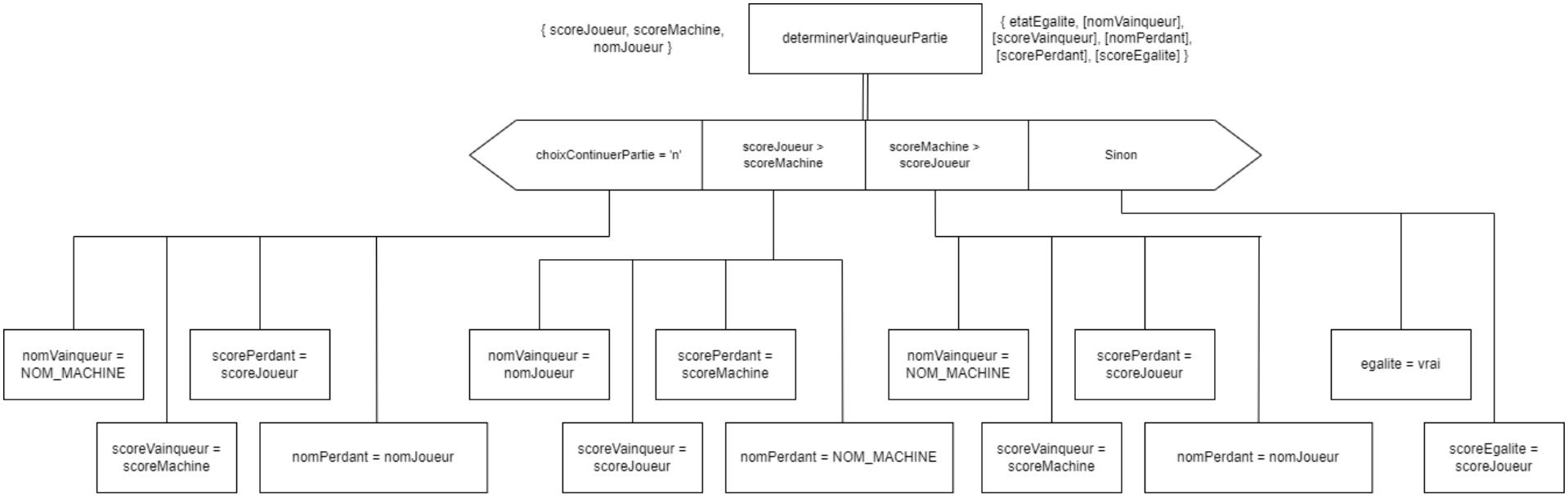
Dictionnaire des variables

Nom	**Type**	**Signification**
scoreJoueur	Nombre entier positif	Le score du joueur tout au long de la partie.
scoreMachine	Nombre entier positif	Le score de la machine tout au long de la partie.
nomJoueur	Chaîne de caractères	Le pseudonyme utilisé par le joueur pour la partie.
NOM_MACHINE	Chaîne de caractères	Nom de la machine.
etatEgalite	Booléen	En cas d'égalité, ce booléen est défini à *true*.
nomVainqueur	Chaîne de caractères	Le nom du vainqueur de la partie.
scoreVainqueur	Nombre entier positif	Le score du vainqueur de la partie.
nomPerdant	Chaîne de caractère	Le nom du perdant de la partie.
scorePerdant	Nombre entier positif	Le score du perdant de la partie.
scoreEgalite	Nombre entier positif	Le score des joueurs en cas d'égalite.
choixContinuerPartie	Caractère	La réponse du joueur à la question de continuer la partie.

determinerVainqueurPartie

Cette étape consiste avant tout à vérifier si le joueur à abandonner ou non. Ensuite, nous comparons le score de la machine avec celui du joueur et déterminons le vainqueur.

Algorithme



Description des sous-problèmes

Cette étape se décomposer en une vérification de trois points :
Si le joueur a abandonné la partie : alors la machine est gagnante et les variables sont définies

Si le score du joueur est supérieur à celui de la machine : alors le joueur à remporté la partie et il est le vainqueur.
Si le score de la machine est supérieur à celui du joueur : alors c'est la machine qui remporte la partie.
Sinon : sinon, les deux scores sont alors à égalité et aucun vainqueur n'est déterminé.

Dictionnaire des variables

Nom	**Type**	**Signification**
scoreJoueur	Nombre entier positif	Le score du joueur tout au long de la partie.
scoreMachine	Nombre entier positif	Le score de la machine tout au long de la partie.
nomJoueur	Chaîne de caractères	Le pseudonyme utilisé par le joueur pour la partie.
NOM_MACHINE	Chaîne de caractères	Nom de la machine.
etatEgalite	Booléen	En cas d'égalité, ce booléen est défini à *true*.
nomVainqueur	Chaîne de caractères	Le nom du vainqueur de la partie.
scoreVainqueur	Nombre entier positif	Le score du vainqueur de la partie.
nomPerdant	Chaîne de caractère	Le nom du perdant de la partie.
scorePerdant	Nombre entier positif	Le score du perdant de la partie.
scoreEgalite	Nombre entier positif	Le score des joueurs en cas d'égalite.

État de finalisation :

Les états de finalisation suivants ont été pris en compte dans la situation algorithmique ainsi quand le code :

Scénario	Pris en compte
Le nombre de manches définit a été atteint.	**Oui**
Abandon du joueur en cours de partie.	**Oui**

Code source

```
/*
 Programme : Jeu de l'equipage
 But : Coder l'algorithme du jeu de l'equipage
 Date de derniere modification : octobre 2023
 Auteurs : E. Desessard & E. Hamid
 Remarques : Code conforme aux specifications internes donnees en cours
*/


#include <iostream>

#include "game-tools.h"


using std::cout;
using std::endl;
using std::cin;

int main()
{

  //VARIABLES

    //Participants

      string nomJoueur; // nom du joueur
      unsigned int scoreJoueur = 0; // score du joueur au long de la partie
      string equipement = "NON_EQUIPE"; // equipement des participants a la partie
      const string NOM_MACHINE = "machine"; // nom de la machine
      unsigned int scoreMachine = 0; // score de la machine au long de la partie

    //Des

      unsigned int valeurBateau; // nombre de points que rapporte l'obtention du bateau
```

```
    unsigned int valeurCapitaine; // nombre de points que rapporte l'obtention du capitaine
    unsigned int valeurEquipage; // nombre de points que rapporte l'obtention de l'equipage

    unsigned short int deUn; // valeur du premier de
    unsigned short int deDeux; // valeur du deuxieme de
    unsigned short int deTrois; // valeur du troisieme de
    unsigned short int deQuatre; // valeur du quatrieme de
    unsigned short int deCinq; // valeur du cinquieme de

  //Manches

    unsigned int nombreManches; // nombre de manches dans la partie
    unsigned int numeroManche; // numero de la manche en cours

  //Lancers

    unsigned int scoreLancer = 0; // score du lancer effectue
    const unsigned short int NOMBRE_LANCERS = 3; // nombre de lancers par manche
    unsigned short int numeroLancer = 1; // numero du lancer en cours

  //FinPartie

    string nomVainqueur; // nom du vainqueur de la partie
    string nomPerdant; // nom du perdant de la partie
    unsigned int scoreVainqueur; // score du vainqueur de la partie
    unsigned int scorePerdant; // score du perdant de la partie
    unsigned int scoreEgalite; // score des deux participants en cas d'egalite

    bool etatEgalite=false; // Etat d'egalite de la partie

  //Autres

    char choixParametrerPartie; // choix du joueur pour parametrer la partie
    char choixContinuerPartie = 'o'; // choix du joueur pour continuer la partie
    char choixContinuerLancers = 'o'; // choix du joueur pour continuer les lancers


// TRAITEMENTS
```

```
// >> initialiserPartie >> nomJoueur, valeurBateau, valeurCapitaine, valeurEquipage, nombreManches

  // >> demanderNomJoueur >> nomJoueur
  cout << "Entrez votre nom de joueur : ";
  cin >> nomJoueur;

  // >> proposerParametrerPartie >> choixParametrerPartie
  cout << "Souhaitez-vous parametrer la partie (o/n) ? : ";
  cin >> choixParametrerPartie;

  // choixParametrerPartie >> parametrerPartie >> valeurBateau, valeurCapitaine, valeurEquipage, nombreManches
  switch (choixParametrerPartie)
  {

  case 'o' : // l'utilisateur souhaite parametrer la partie

    // >> demanderValeurBateau >> valeurBateau
    cout << "Entrez la valeur du bateau : ";
    cin >> valeurBateau;

    // >> demanderValeurCapitaine >> valeurCapitaine
    cout << "Entrez la valeur du capitaine : ";
    cin >> valeurCapitaine;

    // >> demanderValeurEquipage >> valeurEquipage
    cout << "Entrez la valeur de l'equipage : ";
    cin >> valeurEquipage;

    // >> demanderNombreManches >> nombreManches
    cout << "Combien de manches souhaitez-vous jouer ? ";
    cin >> nombreManches;

    pause(2);

    break;

  case 'n' : // l'utilisateur ne souhaite pas parametrer la partie
```

```
    // affectation des valeurs par defaut
    valeurBateau = 600;
    valeurCapitaine = 500;
    valeurEquipage = 400;
    nombreManches = 5;

    cout << "Les valeurs par defaut ont donc ete affectees : " << endl;
    cout << "Valeur du bateau : " << valeurBateau << endl;
    cout << "Valeur du capitaine : " << valeurCapitaine << endl;
    cout << "Valeur de l'equipage : " << valeurEquipage << endl;
    cout << "Nombre de manches : " << nombreManches << endl;

    pause(4);

    break;

  default :

    cout << "choixParametrerPartie est " << choixParametrerPartie << " au lieu de o/n" << endl;

    // en cas d'erreur de saisie, affectation des valeurs par defaut
    valeurBateau = 600;
    valeurCapitaine = 500;
    valeurEquipage = 400;
    nombreManches = 5;

    cout << "Les valeurs par defaut ont donc ete affectees : " << endl;
    cout << "Valeur du bateau : " << valeurBateau << endl;
    cout << "Valeur du capitaine : " << valeurCapitaine << endl;
    cout << "Valeur de l'equipage : " << valeurEquipage << endl;
    cout << "Nombre de manches : " << nombreManches << endl;

    pause(4);

    break;

}
```

```
effacer();

// nomJoueur, valeurBateau, valeurCapitaine, valeurEquipage, nombreLancers >> jouerPartie >> scoreJoueur, scoreMachine

cout << "Debut de la partie :\n" << endl;

pause(1);

  // boucle pour jouer le nombre de manches defini
  for (numeroManche = 1; numeroManche <= nombreManches; numeroManche ++)
  {

    // nomJoueur, valeurBateau, valeurCapitaine, valeurEquipage, numeroManche, choixContinuerPartie, choixContinuerLancers,
    // NOMBRE_LANCERS, numeroLancer, equipement, nombreManches, NOM_MACHINE >> jouerMancheJoueur >> scoreJoueur

      // scoreJoueur, nomJoueur, scoreMachine, NOM_MACHINE >> afficherBilanPartiel >> (ecran)
      cout << "Bilan partiel avant la nouvelle manche : " << endl;
      cout << "Score " << nomJoueur << " : " << scoreJoueur << endl;
      cout << "Score " << NOM_MACHINE << " : " << scoreMachine << "\n" << endl;

      pause(1);

      // numeroManche, nomJoueur>> introduireNouvelleManche >> (ecran)
      cout << "Manche #" << numeroManche << ", " << nomJoueur << endl;

      // boucle pour jouer les 5 lancers
      while (true)
      {
        // vérifie si le joueur souhaite continuer les lancers
        if (choixContinuerLancers == 'n')
        {

          break;

        }

        // numeroLancer, equipement, scoreJoueur, nomJoueur >> jouerLancerJoueur >> scoreJoueur
```

```
// numeroLancer >> introduireNouveauLancer >> (ecran)
cout << "Lancer #" << numeroLancer << " : " << endl;

// >> lancerDes >> deUn, deDeux, deTrois, deQuatre, deCinq
deUn = static_cast<unsigned short int>(random(1, 6));
deDeux = static_cast<unsigned short int>(random(1, 6));
deTrois = static_cast<unsigned short int>(random(1, 6));
deQuatre = static_cast<unsigned short int>(random(1, 6));
deCinq = static_cast<unsigned short int>(random(1, 6));

// deUn, deDeux, deTrois, deQuatre, deCinq >> afficherResultatLancer >> (ecran)
cout << "Contenu du lancer : ";

pause(1);

cout << deUn;

pause(1);

cout << " " << deDeux;

pause(1);

cout << " " << deTrois;

pause(1);

cout << " " << deQuatre;

pause(1);

cout << " " << deCinq << endl;

pause(1);

// deUn, deDeux, deTrois, deQuatre, deCinq, equipement, scoreJoueur >> etablirBilanLancerJoueur >>
```

```
scoreLancer, scoreJoueur, equipement

  // deUn, deDeux, deTrois, deQuatre, deCinq, equipement, scoreJoueur >> calculerEquipementJoueur >>

  equipement
  if (equipement == "NON_EQUIPE") {

    if (deUn == 6 || deDeux == 6 || deTrois == 6 || deQuatre == 6 || deCinq == 6) {

      equipement = "BATEAU"; // ajout du bateau à l'équipement

      scoreLancer += valeurBateau; // ajout de la valeur du bateau au score du lancer

      if (deUn == 5 || deDeux == 5 || deTrois == 5 || deQuatre == 5 || deCinq == 5) {

        equipement = "BATEAU_CAPITAINE"; // ajout du capitaine à l'équipement

        scoreLancer += valeurCapitaine; // ajout de la valeur du capitaine au score du lancer

        if (deUn == 4 || deDeux == 4 || deTrois == 4 || deQuatre == 4 || deCinq == 4) {

          equipement = "BATEAU_CAPITAINE_EQUIPAGE"; // ajout de l'équipage à l'équipement

          scoreLancer += valeurEquipage; // ajout de la valeur de l'équipage au score du lancer

        }

      }

    }

  }

  else {

    if (equipement == "BATEAU") {

      if (deUn == 5 || deDeux == 5 || deTrois == 5 || deQuatre == 5 || deCinq == 5) {
```

```
        equipement = "BATEAU_CAPITAINE"; // ajout du capitaine à l'équipement
        scoreJoueur += valeurCapitaine; // ajout de la valeur du capitaine au score du joueur
        if (deUn == 4 || deDeux == 4 || deTrois == 4 || deQuatre == 4 || deCinq == 4) {
          equipement = "BATEAU_CAPITAINE_EQUIPAGE"; // ajout de l'équipage à l'équipement
          scoreJoueur += valeurEquipage; // ajout de la valeur de l'équipage au score du joueur
        }
      }
    }
    else {
      if (equipement == "BATEAU_CAPITAINE") {
        if (deUn == 4 || deDeux == 4 || deTrois == 4 || deQuatre == 4 || deCinq == 4) {
          equipement = "BATEAU_CAPITAINE_EQUIPAGE";
          scoreJoueur += valeurEquipage;
        }
      }
    }
  }
  // >> calculerScoreLancer >> scoreLancer
  if (equipement == "BATEAU_CAPITAINE_EQUIPAGE") {
```

```
            scoreLancer += (deUn + deDeux + deTrois + deQuatre + deCinq); // ajout de la somme des dés au score du
                                                                          lancer

        }

        // scoreJoueur, scoreLancer >> calculerScoreJoueur >> scoreJoueur
        scoreJoueur += scoreLancer; // ajout du score du lancer du joueur à son score

    // scoreLancer, scoreJoueur, equipement >> afficherBilanLancerJoueur >> (ecran)
    cout << "Bilan du lancer : " << endl;
    cout << "\t-Equipement : " << equipement << endl;
    cout << "\t-Score du lancer : " << scoreLancer << endl;
    cout << "\t-Nouveau score pour " << nomJoueur << " : " << scoreJoueur << endl;

    // réinitialisation du score du lancer
    scoreLancer = 0;

    // vérifie si 3 lancers ont été effectués
    if (numeroLancer >= NOMBRE_LANCERS)
    {

        break;

    }

    numeroLancer = static_cast<unsigned short int>(numeroLancer+1); // incrément du numéro de lancer

  pause(1);

  // >> proposerContinuerLancer >> choixContinuerLancers
  cout << "Souhaitez-vous continuer les lancers (o/n) ? : ";
  cin >> choixContinuerLancers;

  cout << endl;

}
```

```
  // reinitialisation de la variable equipement à la fin du tour du joueur
  equipement = "NON_EQUIPE";

  // reinitialisation de la variable numeroLancer à la fin du tour du joueur
  numeroLancer = static_cast<unsigned short int>(1);

  // >> proposerContinuerPartie >> choixContinuerPartie
  cout << "Souhaitez-vous continuer la partie (o/n) ? : ";
  cin >> choixContinuerPartie;

pause(1);
effacer();

// choixContinuerPartie >> verifierContinuerPartie >>
if (choixContinuerPartie == 'n')
{

  break;

}

// nomJoueur, valeurBateau, valeurCapitaine, valeurEquipage, numeroManche, NOMBRE_LANCERS, numeroLancer, equipement,
// nombreManches, NOM_MACHINE >> jouerMancheMachine >> scoreMachine

  // scoreJoueur, nomJoueur, scoreMachine, NOM_MACHINE >> afficherBilanPartiel >> (ecran)
  cout << "Bilan partiel avant la nouvelle manche : " << endl;
  cout << "Score " << nomJoueur << " : " << scoreJoueur << endl;
  cout << "Score " << NOM_MACHINE << " : " << scoreMachine << "\n" << endl;

  pause(1);

  // numeroManche, NOM_MACHINE>> introduireNouvelleManche >> (ecran)
  cout << "Manche #" << numeroManche << ", " << NOM_MACHINE << endl;

  // boucle pour jouer les 5 lancers
  while (true)
  {
```

```
// numeroLancer, equipement, scoreMachine, NOM_MACHINE >> jouerLancerMachine >> scoreMachine

  // numeroLancer >> introduireNouveauLancer >> (ecran)
  cout << "Lancer #" << numeroLancer << " : " << endl;

  // >> lancerDes >> deUn, deDeux, deTrois, deQuatre, deCinq
  deUn = static_cast<unsigned short int>(random(1, 6));
  deDeux = static_cast<unsigned short int>(random(1, 6));
  deTrois = static_cast<unsigned short int>(random(1, 6));
  deQuatre = static_cast<unsigned short int>(random(1, 6));
  deCinq = static_cast<unsigned short int>(random(1, 6));

  // deUn, deDeux, deTrois, deQuatre, deCinq >> afficherResultatLancer >> (ecran)
  cout << "Contenu du lancer : ";

  pause(1);

  cout << deUn;

  pause(1);

  cout << " " << deDeux;

  pause(1);

  cout << " " << deTrois;

  pause(1);

  cout << " " << deQuatre;

  pause(1);

  cout << " " << deCinq << endl;

  pause(1);
```

```
// deUn, deDeux, deTrois, deQuatre, deCinq, equipement, scoreMachine >> etablirBilanLancerMachine >>
scoreLancer, scoreMachine, equipement

  // deUn, deDeux, deTrois, deQuatre, deCinq, equipement, scoreMachine >> calculerEquipementMachine >>
  equipement
if (equipement == "NON_EQUIPE") {

  if (deUn == 6 || deDeux == 6 || deTrois == 6 || deQuatre == 6 || deCinq == 6) {

    equipement = "BATEAU"; // ajout du bateau à l'équipement

    scoreLancer += valeurBateau; // ajout de la valeur du bateau au score du lancer

    if (deUn == 5 || deDeux == 5 || deTrois == 5 || deQuatre == 5 || deCinq == 5) {

      equipement = "BATEAU_CAPITAINE"; // ajout du capitaine à l'équipement

      scoreLancer += valeurCapitaine; // ajout de la valeur du capitaine au score du lancer

      if (deUn == 4 || deDeux == 4 || deTrois == 4 || deQuatre == 4 || deCinq == 4) {

        equipement = "BATEAU_CAPITAINE_EQUIPAGE"; // ajout de l'équipage à l'équipement

        scoreLancer += valeurEquipage; // ajout de la valeur de l'équipage au score du lancer

      }

    }

  }

}

else {

  if (equipement == "BATEAU") {

    if (deUn == 5 || deDeux == 5 || deTrois == 5 || deQuatre == 5 || deCinq == 5) {
```

```
            equipement = "BATEAU_CAPITAINE"; // ajout du capitaine à l'équipement
            scoreMachine += valeurCapitaine; // ajout de la valeur du capitaine au score de la machine
            if (deUn == 4 || deDeux == 4 || deTrois == 4 || deQuatre == 4 || deCinq == 4) {
              equipement = "BATEAU_CAPITAINE_EQUIPAGE"; // ajout de l'équipage à l'équipement
              scoreMachine += valeurEquipage; // ajout de la valeur de l'équipage au score de la machine
            }
          }
        }
        else {
          if (equipement == "BATEAU_CAPITAINE") {
            if (deUn == 4 || deDeux == 4 || deTrois == 4 || deQuatre == 4 || deCinq == 4) {
              equipement = "BATEAU_CAPITAINE_EQUIPAGE";
              scoreMachine += valeurEquipage;
            }
          }
        }
    }

    // >> calculerScoreLancer >> scoreLancer
    if (equipement == "BATEAU_CAPITAINE_EQUIPAGE") {
```

```
      scoreLancer += (deUn + deDeux + deTrois + deQuatre + deCinq); // ajout de la somme des dés au score du lancer

    }

    // scoreMachine, scoreLancer >> calculerScoreMachine >> scoreMachine
    scoreMachine += scoreLancer; // ajout du score du lancer du machine à son score

    // scoreLancer, scoreJoueur, equipement >> afficherBilanLancerJoueur >> (ecran)
    cout << "Bilan du lancer : " << endl;
    cout << "\t-Equipement : " << equipement << endl;
    cout << "\t-Score du lancer : " << scoreLancer << endl;
    cout << "\t-Nouveau score pour " << NOM_MACHINE << " : " << scoreMachine << endl;

    pause(1);

    // réinitialisation du score du lancer
    scoreLancer = 0;

    // vérifie si 3 lancers ont été effectués
    if (numeroLancer >= NOMBRE_LANCERS)
    {

      break;

    }

    numeroLancer = static_cast<unsigned short int>(numeroLancer+1); // incrément du numéro de lancer

    cout << endl;

  }

  // reinitialisation de la variable equipement à la fin du tour de la machine
  equipement = "NON_EQUIPE";

  // reinitialisation de la variable numeroLancer à la fin du tour de la machine
  numeroLancer = static_cast<unsigned short int>(1);
```

```
    pause(1);

    // >> proposerContinuerPartie >> choixContinuerPartie
    cout << "Souhaitez-vous continuer la partie (o/n) ? : ";
    cin >> choixContinuerPartie;

  pause(2);
  effacer();

    // choixContinuerPartie >> verifierContinuerPartie >>
    if (choixContinuerPartie == 'n')
    {

      break;

    }
    else {

      choixContinuerLancers = 'o';

    }

  }

// scoreJoueur, scoreMachine, nomJoueur >> finaliserPartie >>

  // scoreJoueur, scoreMachine, nomJoueur >> determinerVainqueurPartie >> etatEgalite,
  // nomVainqueur, scoreVainqueur, nomPerdant, scorePerdant, [scoreEgalite]
  if (choixContinuerPartie == 'n')
  {

    nomVainqueur = NOM_MACHINE;
    scoreVainqueur = scoreMachine;

    nomPerdant = nomJoueur;
    scorePerdant = scoreJoueur;

  }
```

```
else {

  if (scoreJoueur > scoreMachine)
  {

    nomVainqueur = nomJoueur;
    scoreVainqueur = scoreJoueur;

    nomPerdant = NOM_MACHINE;
    scorePerdant = scoreMachine;

  }
  else if (scoreMachine > scoreJoueur)
  {

    nomVainqueur = NOM_MACHINE;
    scoreVainqueur = scoreMachine;

    nomPerdant = nomJoueur;
    scorePerdant = scoreJoueur;

  }
  else {

    scoreEgalite = scoreJoueur;

    etatEgalite = true;

  }

}

// nomVainqueur, scoreVainqueur, nomPerdant, scorePerdant, [scoreEgalite] >> afficherVainqueur >> (ecran)
cout << "Resultats de la partie :\n" << endl;

pause(2);

if (choixContinuerPartie == 'n')
```

```
    {

      cout << nomVainqueur << " a gagne la partie par abandon de " << nomPerdant << endl;

    }
    else {

      if (etatEgalite == true)
      {

        cout << "Les deux joueurs sont a egalite a " << scoreEgalite << " points" << endl;

      }
      else {

        if (scoreVainqueur == scoreJoueur)
        {

          cout << "Bravo " << nomVainqueur << ", votre score : " << scoreVainqueur << " > au score de " << nomPerdant << " : " << scorePerdant << endl;

        }
        else {

          cout << "Dommage " << nomPerdant << ", votre score : " << scorePerdant << " < au score de " << nomVainqueur << " : " << scoreVainqueur << endl;

        }

      }

    }

    pause(5);

  return 0;

}
```