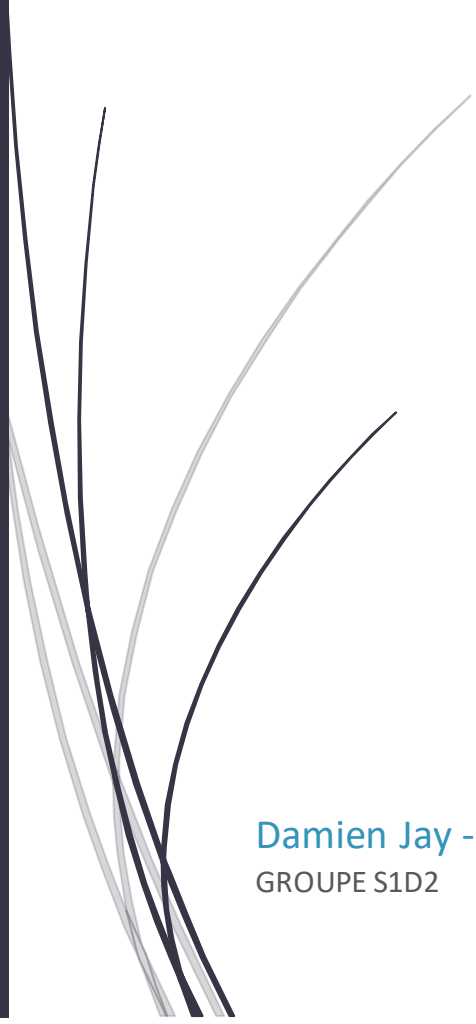
20/12/2017

RAPPORT DE PROJET ALGORITHMIQUE

Civilization 0



Damien Jay - Lise KABBACHE – Hédhie Krouk – Jean Rassemusse
GROUPE S1D2

Damien Jay Lise Kabbache Hédhie Krouk Jean Rassemusse

Présentation du Jeu :

Pour notre jeu, nous avons décidé ensemble de choisir comme thème « ANIMALS WAR - Les animaux moche vs. Les animaux beaux ». Nous avons choisi ce thème car il nous a paru original, intéressant et drôle. Ainsi, nous disposons de 5 familles d'animaux : les Primates, les Poissons, les Gallinacés, les Rongeurs et les Amphibiens. Ces 5 familles ont chacune des noms d'unités, de bâtiments et d'ennemis en lien avec leur famille.

	Nom Tribut	Primates	Poissons	Gallinacés	Rongeurs	Amphibiens
Villes	Capitale	TENOCHTITLAN	ATLANTIS	LUTECE	HAMELIN	DULVEY
	Colonie	Mingun	Ys	Massilia	Orlova	Lhassa
Unités	Soldat	Aye-Aye	Rascasse	Pintade	Rat-Taupe-Nu	Cécilie
	Canon	Nasique	Poisson Ogre	Dindon	Castor	Crapaud Buffle
	Cavalerie	Singe Hurleur	Mérou	Talégalle	Ragondin	Axolotl
Bâtiments	Ferme	Forêt de Papayers	Champ d'Algues	Champ de Maïs	Potager	Ferme aux Moustiques
	Mine	Mine de la Cascade	Mine sous-marine	Silo	Moulin	Creux
	Carrière	Temple Caché	Cavité	Champ de Tournesol	Stère	Parterre de Mousse
	Caserne	Baobab géant	Barrière de Corail	Couvoir	Barrage	Mare
	Etable	Forêt d'Acajous	Colonie d'Eponges	Grenier	Tanière	Bauge
	Marché	Marché de Tenochtitlan	Marché d'Atlantis	Marché de Lutèce	Marché d'Hamelin	Marché du Bayou
Unités Ennemies	Soldat Ennemi	Saïmiri	Poisson Clown	Faisan	Octodon	Rainette verte
	Canon Ennemi	Maki Catta	Poisson Perroquet	Coq	Porc-épic	Dendrobate Azurée
	Cavalerie Ennemie	Gibbon	Hippocampe	Paon	Capybara	Salamandre Tachetée
Bonus Bâtiments	Bonus Ferme	Papaye géante	Algue géante	Maïs géant	Citrouille géante	Moustique géant
	Bonus Mine	Diamant	Rubis	Grain fossilisé	Farine extra-fine	Quartz
	Bonus Carrière	Pierre polie	Cobalt	Graine de tournesol géante	Souche	Lichen
	Bonus Caserne	Branche de Baobab	Polype	Couveuse	Rondin	Nénuphar
	Bonus Etable	Noix de cajou	Coquillage	Fragment de plancher intact	Racine rare	Ver doré

Damien Jay Lise Kabbache Hédhie Krouk Jean Rassemusse

Liste des améliorations choisies :

Faciles :

- Un écran de titre : Votre jeu devra comporter un écran de titre raisonnable.
- Faites entrer la cavalerie : Ajoutez un nouveau type d'unités : la cavalerie qui ne peut être achetée que si la caserne est au moins au niveau 2.
- Les poules d'Abigail : Ajoutez un nouveau type de bâtiment : l'étable augmentant la production de nourriture dans la ville et dont le niveau ne peut pas dépasser celui de la ferme.

Intermédiaires :

- Oui monseign'Or : Ajoutez un nouveau bâtiment : le marché produisant de l'or nécessaire pour acheter des troupes.
- Universalité : Le joueur doit pouvoir choisir entre 5 civilisations déférentes.
- Ressources bonus : Chaque ville a accès aléatoirement à des ressources bonus (blé, charbon, ...) permettant la construction de version amélioré des bâtiments (ferme, mine...).

Avancées :

- Colonisation : Ajoutez la possibilité de construire un colon. Ce colon peut être utilisé pour fonder une deuxième ville que le joueur pourra gérer comme la capitale.

Expertes :

- A deux c'est encore mieux : Ajoutez un mode Hot Seat (deux joueurs sur 1 ordinateur) à votre jeu.

Pour notre jeu, nous avons utilisé 5 unités :

- GestionEcran : Nous avons utilisé cette unité afin de pouvoir créer des bordures et déplacer le curseur à notre guise.
- unitAffichage : A l'intérieur de cette unité, nous avons 11 procédures qui permettent de gérer l'affichage de tous les cadres, et d'afficher toutes les informations nécessaires à l'écran dans une bonne disposition. Pour cela, nous avons besoin de l'unité GestionEcran et l'unité unitType.
- unitGestionCombat : Dans cette unité, nous avons besoin des unités unitAffichage, unitType et GestionEcran car elles nous permettent d'avoir les informations nécessaires afin de gérer les combats et le recrutement d'unité, ainsi que l'affichage. Il y a donc à l'intérieur de cette unité 2 procédures : ecranMilitaire et ecranCombat qui permettent de gérer le choix du joueur dans l'écran de combat et qui gère également toutes les possibilités suite à ce choix.
- unitGestionGeneral : Dans cette unité, nous avons besoin des unités GestionEcran, unitGestionCombat, unitAffichage et unitType. Nous avons 6 procédures et une fonction à l'intérieur de cette unité. La fonction sert à initialiser les données du tableau. Les procédures servent à lance une partie ou pour fermer le jeu, gérer sa

civilisation, gérer le mécanisme de fin de tour, enregistrer le choix de civilisation du joueur, initialiser les variables lors d'une nouvelle partie et de gérer sa capitale.

- unitType : A l'intérieur de cette unité, nous avons créé tous les types nécessaires pour ce projet, c'est-à-dire que nous avons déclaré une série de type Record, chacun correspondant à un aspect du jeu.

Algorithme d'une procédure de notre programme :

Nom : ecranCapitale

Rôle : Le programme va appeler la procédure d'affichage correspondant à l'écran de la capitale et aussi gérer le choix du joueur concernant les bâtiments et le fait de rester sur l'écran ou de revenir à l'écran principal.

Glossaire :

I. Paramètres en Entrée :

- **civ :** variable de type civilisation, un tableau correspondant à la civilisation choisie par le joueur et contenant tous ses attributs (nom, capitale, noms et niveaux des bâtiments …),
- **cdn :** variable de type condition, un record contenant des booléens correspondant au souhait du joueur d'arrêter ou non le jeu, au relancement d'une procédure tant que le joueur ne la quitte pas ou encore signifiant qu'une partie est en cours ou non… Dans cette procédure, la variable utilisée est cdn.check qui est égale à VRAI lorsque le joueur souhaite quitté un menu et à FAUX tant que le joueur y reste.
- **tours :** variable de type entier qui s'incrémente de 1 à chaque fois que le joueur choisi de terminer son tour.

II. Paramètres en sortie :

- Aucun

III. Paramètres en entrée-sortie :

- **j :** variable de type joueur, un record contenant tous les attributs relatifs à la partie du joueur (l'effectif de son armée, ses ressources, la variable relative à ses choix, …) la variable de ce record qui sera utilisée dans cette procédure est : j.choixJoueur, une variable entière qui sera saisie au clavier qui correspond au choix d'action du joueur.

IV. Variables et constantes de travail :

- **erreurs :** variable de type erreur, un record contenant toutes les erreurs, pouvant arriver lors d'une partie, ici seront utilisées :
  - construct, variable booléenne qui passe à vrai si le joueur veut lancer une construction alors qu'une autre est déjà en cours,
  - nivMax, variable booléenne qui passe à vrai lorsqu'un bâtiment atteint le niveau maximal
  - Etable, variable booléenne qui passe à vrai si le joueur veut lancer la construction de l'étable sans avoir une ferme de niveau suffisant

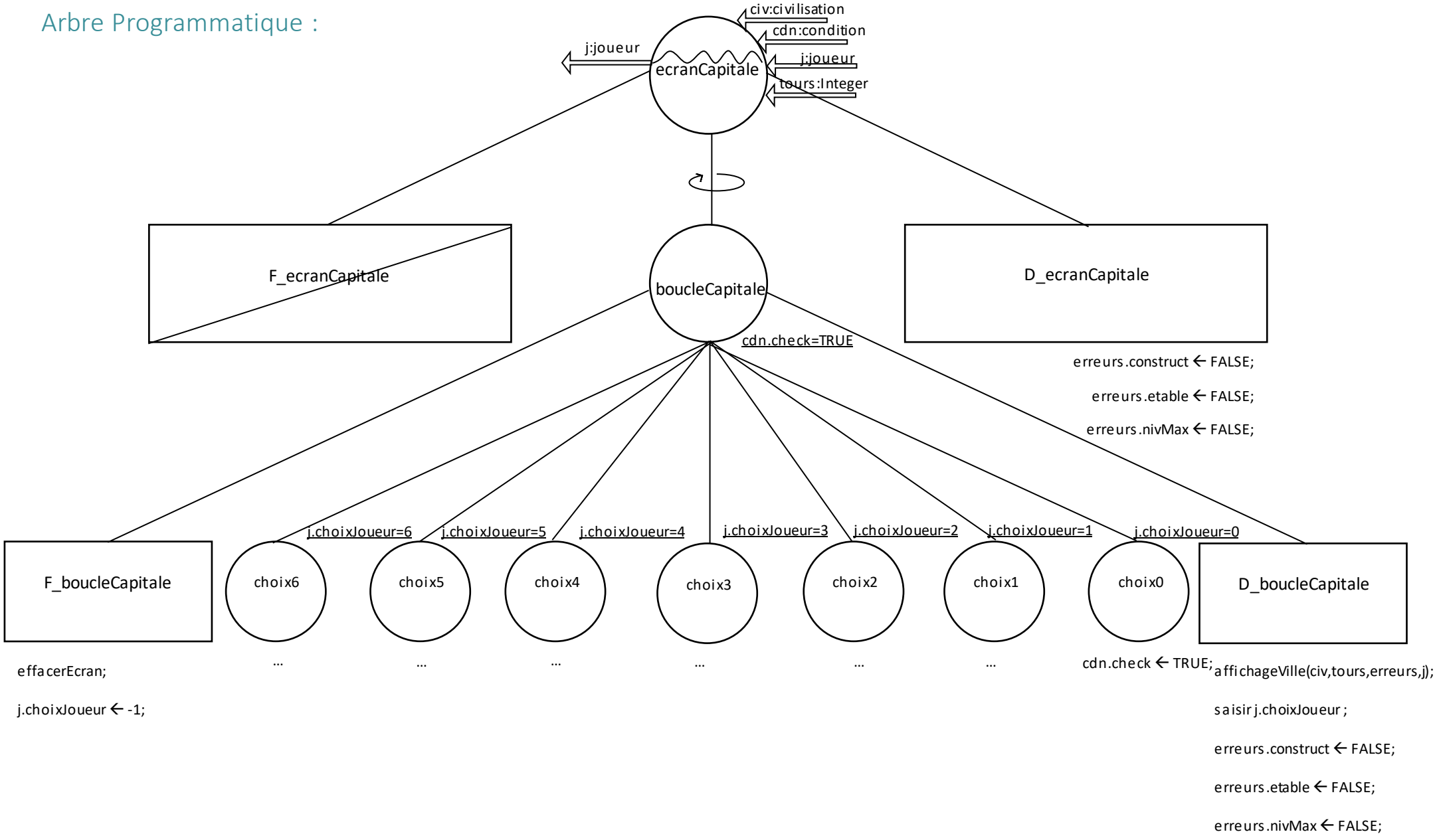
Principe : Dès l'appel de cette procédure, ecranCapitale va appeler la procédure d'affichage correspondante afin d'offrir au joueur une interface où il pourra se retrouver. Tant qu'il n'aura pas quitté l'écran, c'est-à-dire, tant que cdn.check sera fausse, l'utilisateur pourra saisir des entiers chacun correspondant à une des 7 actions disponibles,

construire une ferme (saisir 1), une mine (saisir 2), une carrière (saisir 3), une caserne (saisir 4), une étable (saisir 5) et un marché (saisir 6), ou quitter l'écran de capitale (saisir 0). Lorsque le joueur choisit de construire (ou améliorer) un des bâtiments, le programme va tout d'abord vérifier qu'aucun autre bâtiment n'est en cours de construction. Si c'est le cas, erreurs.construct passe à vrai et un message d'erreur sera affiché à l'aide de la procédure d'affichage à la prochaine itération. Dans le cas contraire, il va vérifier si le bâtiment n'est pas déjà au niveau 3 ou à 4 si le joueur possède la ressource bonus correspondant au bâtiment concerné. Si le bâtiment est déjà au niveau maximum, erreurs.nivMax passera à vrai et un message d'erreur sera affiché à l'aide de la procédure d'affichage à la prochaine itération. Dans le cas contraire, la construction ou l'amélioration du bâtiment sera lancée. Si le bâtiment que le joueur souhaitait améliorer était l'étable, en saisissant 5, le programme va tout d'abord vérifier si le niveau de la ferme est supérieur ou inférieur à celui de l'étable. Si l'étable est de niveau supérieur à la ferme, erreurs.etable pas sera à vrai et un message d'erreur s'affichera à l'aide de la procédure d'affichage à la prochaine itération. Dans le cas contraire, le programme procédera comme pour les autres bâtiments. Si le joueur saisit 0, il quittera l'écran puisque cdn.check passera à vrai.

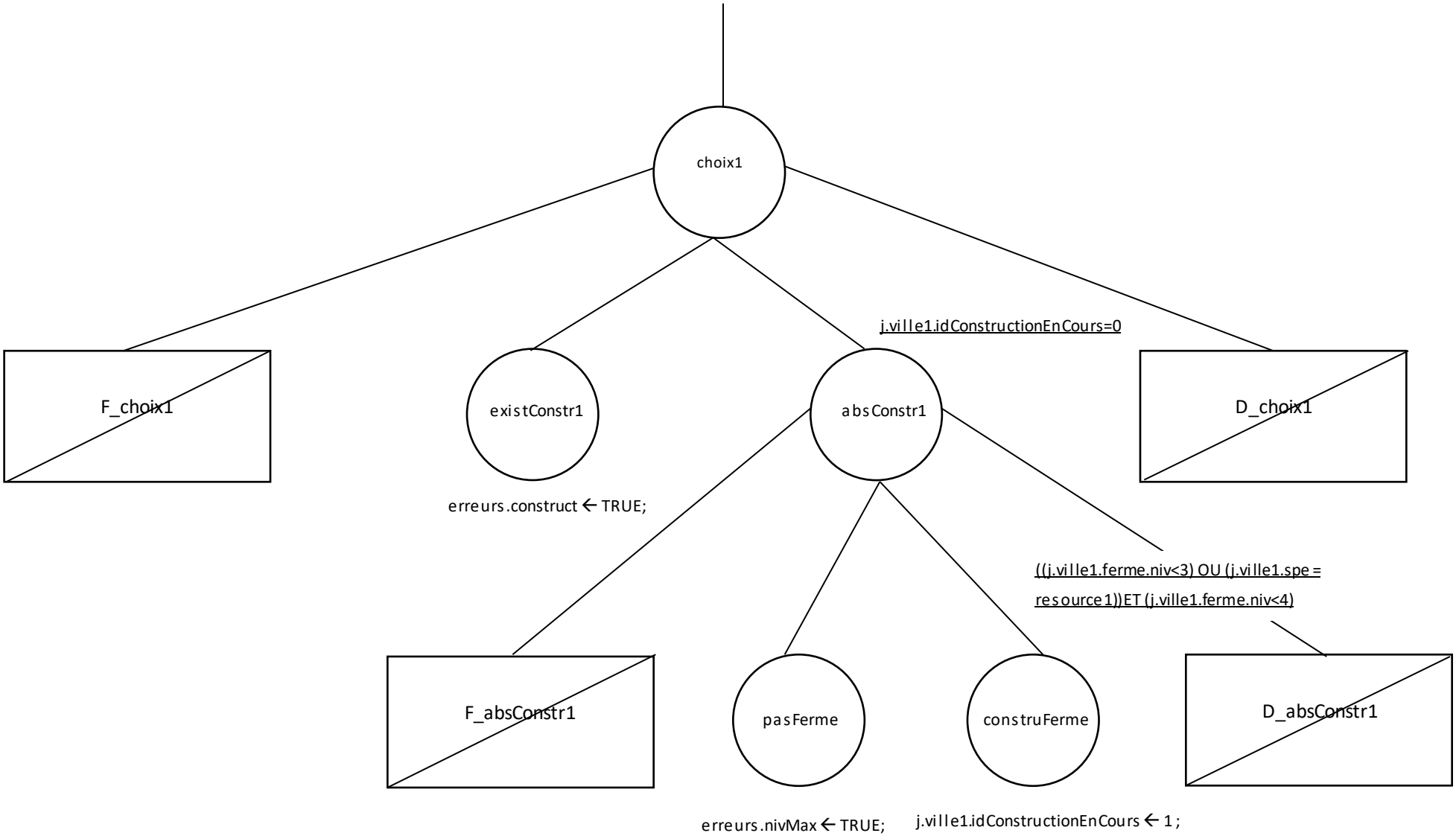
Jeux d'essais :

cdn.check	erreurs.construct	erreurs.etable	erreurs.nivMax	j.choixJoueur	j.ville1.idConstructionEncours
~~FALSE~~	~~FALSE~~	~~FALSE~~	~~FALSE~~	~~2~~	~~1~~
TRUE	~~FALSE~~	~~FALSE~~	~~FALSE~~	~~5~~	1
	FALSE	~~TRUE~~	FALSE	0	
		FALSE			

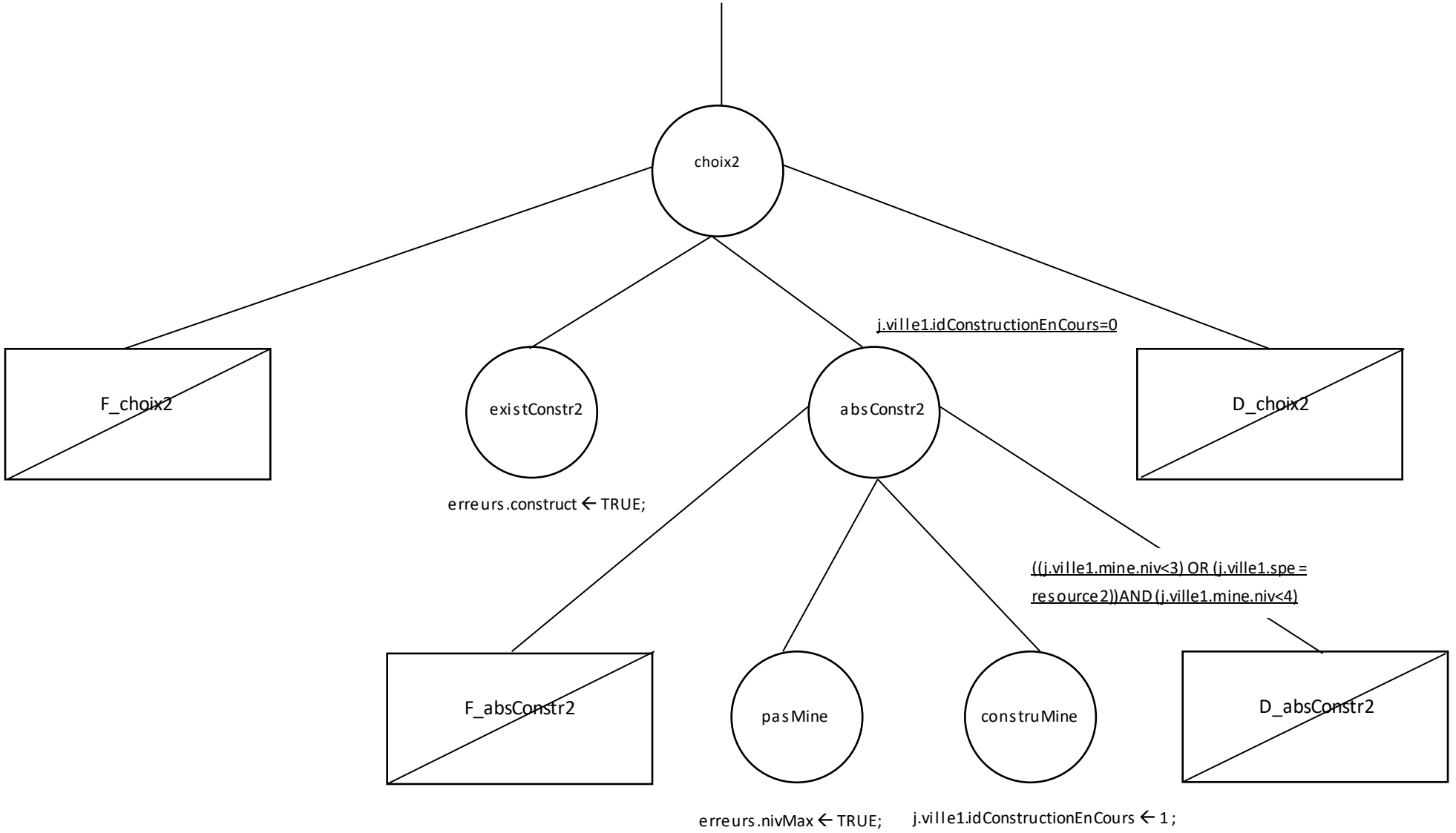
Arbre Programmatique :



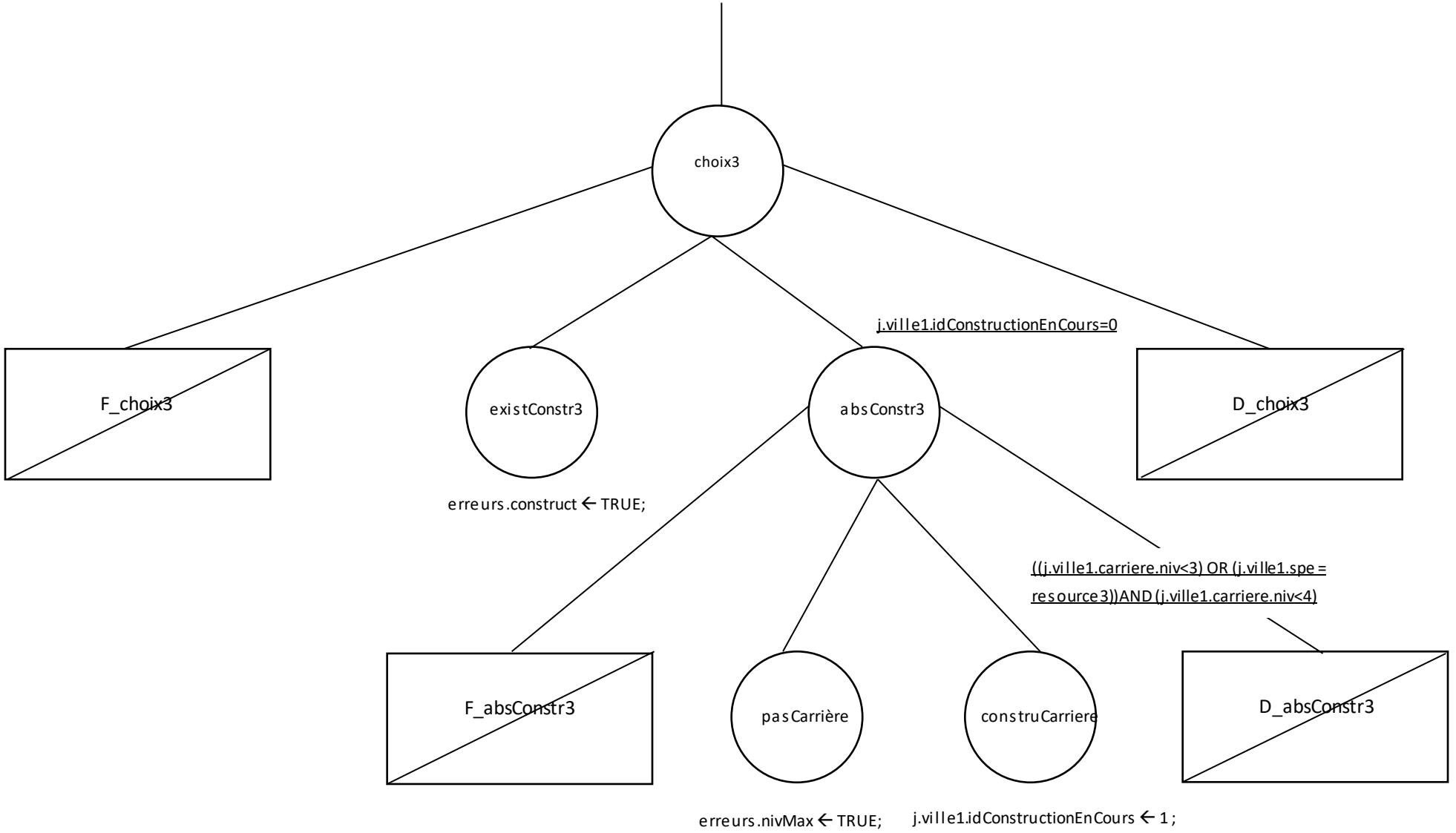
Damien Jay Lise Kabbache Hédhie Krouk Jean Rassemusse



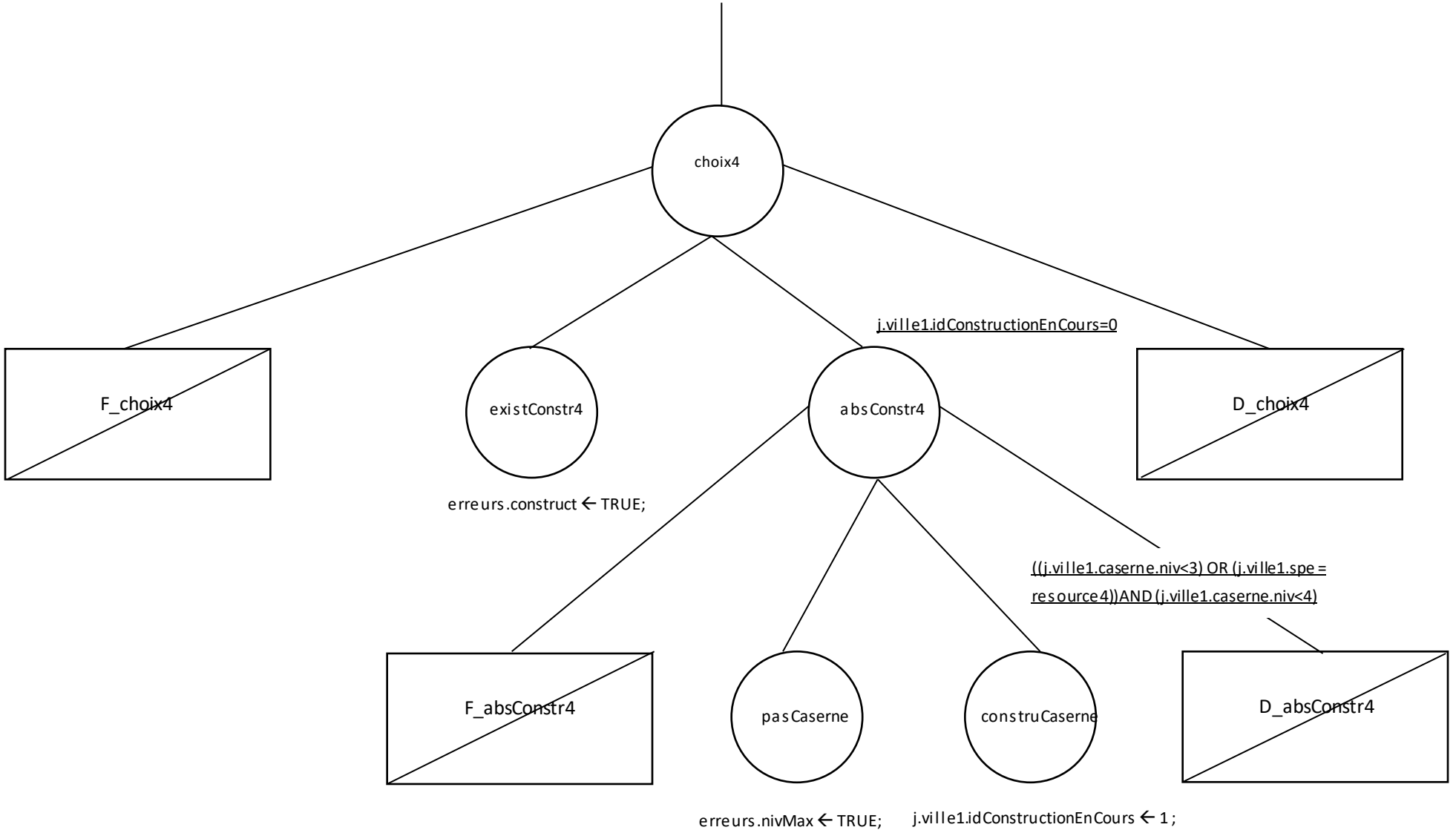
Damien Jay Lise Kabbache Hédhie Krouk Jean Rassemusse



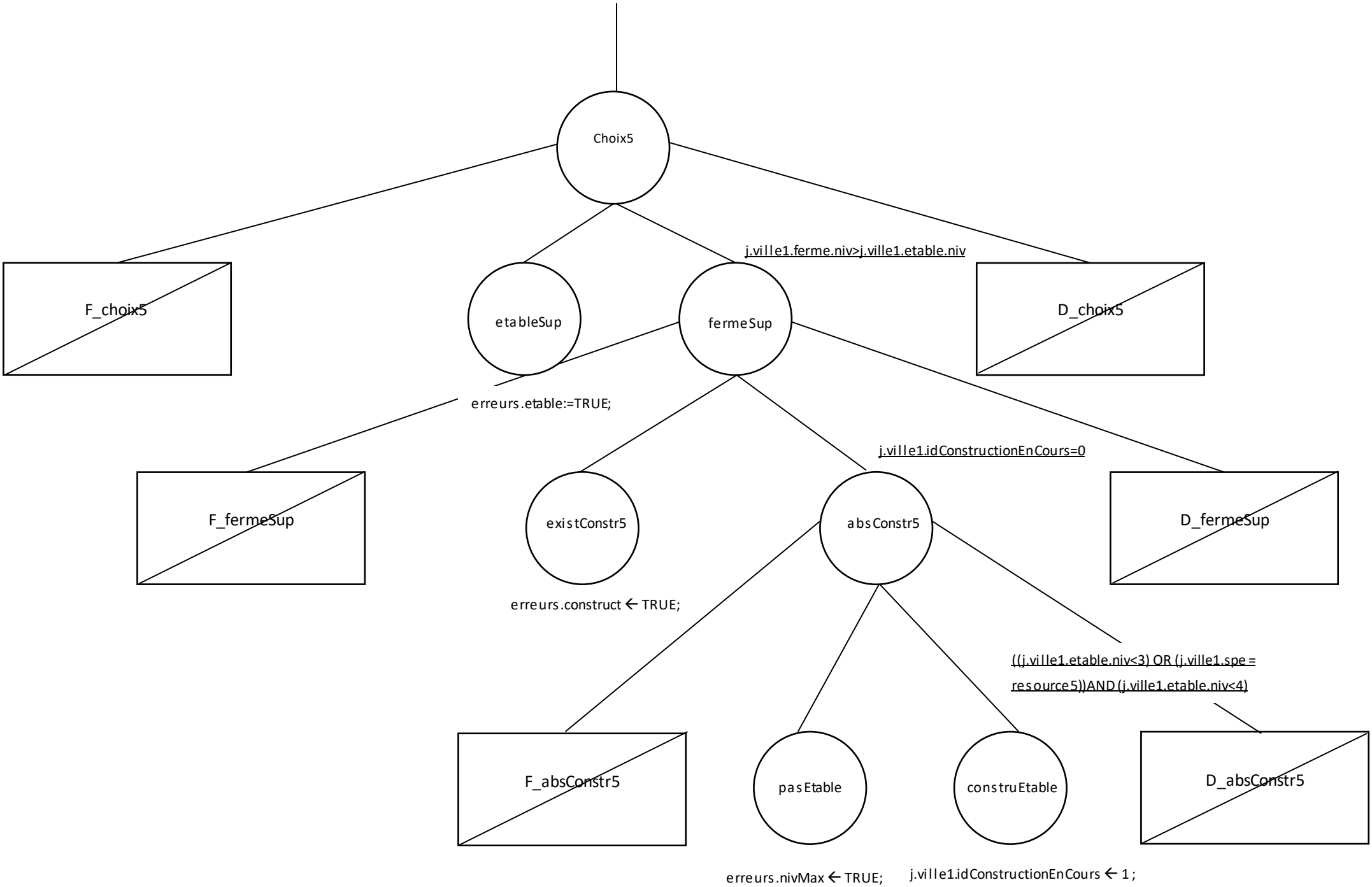
Damien Jay Lise Kabbache Hédhie Krouk Jean Rassemusse



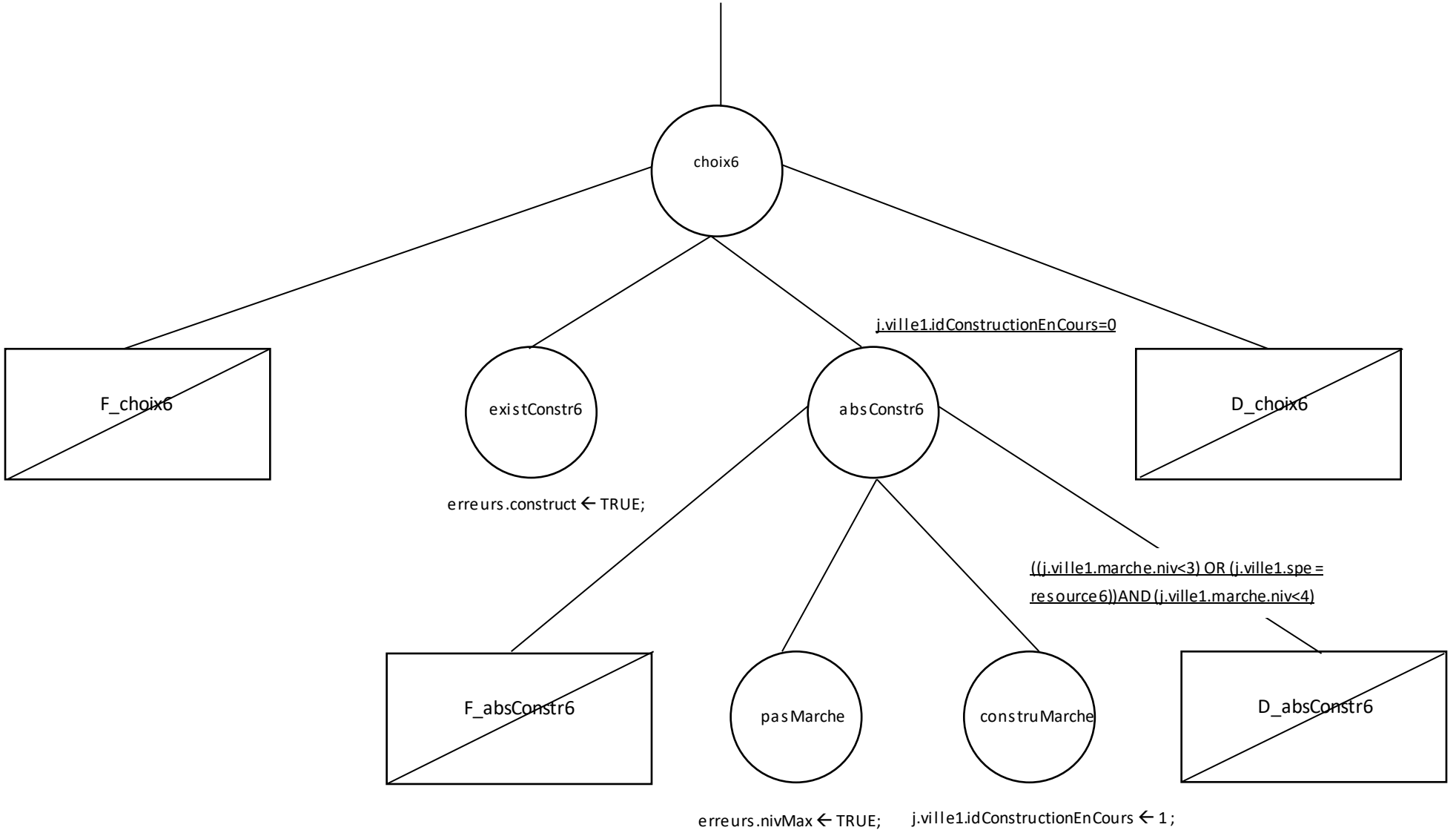
Damien Jay Lise Kabbache Hédhie Krouk Jean Rassemusse



Damien Jay Lise Kabbache Hédhie Krouk Jean Rassemusse



Damien Jay Lise Kabbache Hédhie Krouk Jean Rassemusse



Validé