RefCard

CLEAN CODE : NOMMAGE

Michelle Avomo & Dorian Bussi

Mai 2018

Cabinet de conseil IT et Agilité | SOAT

REFCARD

CLEAN CODE : Nommage

Version : Mai 2018

- Le nommage est la première chose qu'un **développeur apprend.**
  C'est aussi la première chose qu'il **oublie.**

- **Rendre son code lisible** c'est faciliter sa compréhension.
  Comprendre le code c'est pouvoir le maintenir.
  Pouvoir maintenir du code c'est **répondre au besoin du client.**

COMMUNIQUER L'INTENTION

est la règle de base du nommage

- Chaque nom de classe, interface, abstraction, méthode ou variable doit être aussi **clair** que possible sur ce qu'elle fait.

- Un bon nommage répond à la question quoi et pas à comment.

Check-list

Nous écrivons du code d'abord **pour les humains** et **non pour les machines !**

Le nommage est un sport d'équipe

- ✓ Toujours **discuter** des conventions à adopter avec ses pairs
- ✓ **Challenger** l'existant et/ou proposer une **harmonisation** du nommage lorsqu'il n'y en a pas

Toujours communiquer l'intention

- ✓ Les noms que nous assignons aux variables, méthodes, classes, etc. sont nos **outils de communication**
- ✓ Un nom explicite permet de **comprendre** ce que fait le code sans lire toute l'implémentation
- ✓ Du code communiquant **clairement son intention** se passe de commentaires

Privilégier la communication

- ✓ Des noms prononçables permettent de **fluidifier la communication** au sein de l'équipe
- ✓ Des noms recherchables (donc distinctifs) permettent de **gagner du temps**

Le scope influence la taille des noms "recommandables"

- ✓ Les noms des variables et des méthodes devraient toujours être **aussi courts que possible, aussi longs que nécessaire**.

CLEAN CODE : Nommage

Des noms faciles à lire et à prononcer
permettent des conversations fluides

Exemples à éviter	Ne pas obscurcir	Exemples recommandés
Nbr2Cdt, EtpRtmt	Choisir un nom **imprononçable**	nombreDeCandidats, etapeDuRecrutement
n2CVrc, nbRec	Convention **subjective** (ex : abréviation)	nbCVRecus, totalCvRecus
p_strName, _nom	Utiliser une **notation désuète** (ex : hongroise)	nom, nomDuCandidat

Des noms aux concepts implicites
complexifient la compréhension fonctionnelle

Exemples à éviter	Ne pas désinformer	Exemples recommandés
annee, diplome	Nommer selon un **niveau d'abstraction inadéquat**	experienceDuCandidat, anneeDuDiplome
souhaits, missions	Garder un nom alors que **le sens a évolué**	formationsProposees, preferencesDeMission
candidatSet, formateursMap	Choisir un nom qui mène à de **fausses conclusions**	groupeDeCandidats, formateursMasterClass

Des noms similaires
non distinctifs prêtent à confusion

Exemples à éviter	Ne pas porter à confusion	Exemples recommandés
candidatAC, candidatA	Noms similaires, **peu distinctifs**	candidatAContacter, candidatAccepte
candidatInfo, candidatData	Mots **parasites**	adresse, profil, adresseCandidat

Des noms rappelant le type ou numérotés
n'apportent aucune information

Exemples à éviter	Ne pas se répéter	Exemples recommandés
candidatsList, candidatsMap,	Annoncer **le type** dans le nom	candidats, candidatsParTechno
p1, p2	**Numéroter** les variables	premiereProposition, contreProposition

Conventions de nommage

Quelques règles usuelles et bonnes pratiques

Élément	Convention	Exemple
Classe / interface	Noms / groupes nominaux Eviter autant que possible les mots parasites (manager, service...)	Recrutement, ConsultantRecruteur, ExperiencesCandidat
Variable de classe (field)	Noms / groupes nominaux	nom, adresse, preferencesDeMission
Variable locale (variable)	Noms / groupes nominaux Idéalement dérivé du nom de la classe	consultantsSurParis, clientActuelDuConsultant
Booléen	Prédicat d'état : verbe être + complément	estAccepte, estRefuse, estEnMission
Fonctions / méthode	Verbes à l'impératif Si retournant un booléen : Prédicat	annulerEntretien(), enregistrer(), estCoopte()
Enumération / type fermé	Adjectif ou nom	EN_COURS, ACCEPTE
Événement	Verbe au participe passé	integrationTerminee, cooptationValidee, missionChoisie

Une convention de nommage par langage de programmation est recommandée.

Choisissez-la en équipe !

Attention :

Ces règles peuvent ne pas être adaptées pour les circonstances suivantes :

Autres choix d'équipe

Programmation fonctionnelle

DDD - Ubiquitous language

Privilégier la lisibilité

Nommage et taille optimale

Un code propre est simple à lire.
Il se lit toujours « *Comme une prose bien écrite* » selon Grady Booch.

Les noms des variables et des méthodes devraient toujours **être aussi courts que possible, aussi longs que nécessaire.**	
Trop longs	nombreDeCvRecusDepuisLaDernierePromotion formationsProposeesAuxSoatiensEnMissionDePlusDeDixHuitMois
Trop courts	ndcv, cvs, f, fp, fps, enMission, dixHuit
Recommandables	nbCvRecusDepuisDernierePromo, formationsProposees, formationsInternesPourMissionsLongues

The "scope length rule"

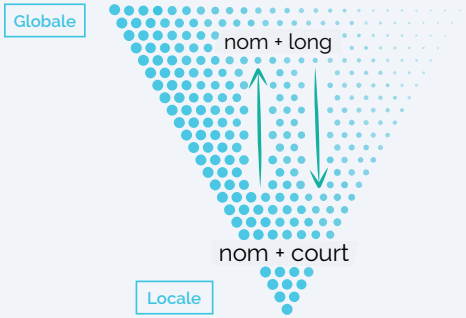
La règle du scope prend en compte la portée des variables et des méthodes pour déterminer leur taille « recommandable »

- **Pour une variable :** Plus sa portée est grande, plus son nom doit être explicite (nom de variable plutôt long).

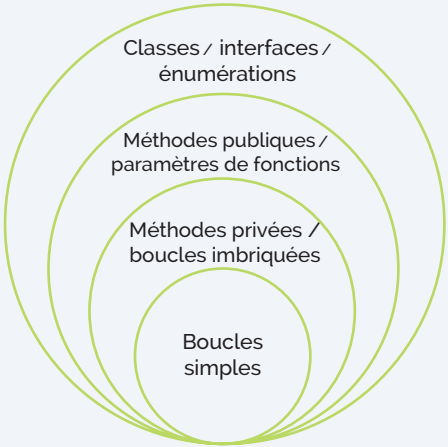
- **Pour une méthode/une classe :** Plus elle est exposée, plus son nom devrait être générique, donc court.

Variable

Peut être manipulée dans plusieurs contextes différents.
Son nommage est explicite quel que soit le contexte.

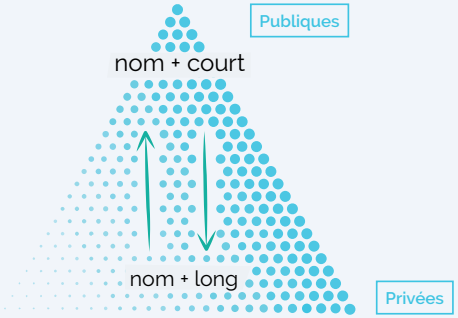


Le contexte est suffisamment petit et auto-explicatif.



Classe / fonction

Peut intégrer plusieurs opérations et doit
masquer sa complexité interne vis-à-vis de ses appelants.



Fait une opération très précise qui devrait être révélée par son nommage.

MICHELLE AVOMO - SOAT

Développeuse Full Stack, je suis curieuse et naturellement enthousiaste. Chez SOAT, je participe depuis 2017 à la montée en compétence des Soatiens sur les outils du Craft (Tests, Refacto, etc.).
Chez mes clients, au-delà de répondre au besoin, j'aime apprendre et transmettre. Très souvent partante pour une conférence, vous me retrouverez dans des Meetups (souvent Crafts ou Code Retreat) sur Paris.

DORIAN BUSSI - SOAT

Développeur curieux, j'aime comprendre les fondamentaux des outils et pratiques qui me servent au quotidien. Je suis donc naturellement attiré par les pratiques de tests et le courant Software Craftsmanship. J'explore l'adaptation des bonnes pratiques à la lumière de nouvelles idées. Je participe ainsi à la vie de SOAT au travers de sa communauté Craft. Et vous pourrez me croiser dans les Meetups parisiens qui y sont consacrés.

SOAT
89 quai Panhard et Levassor 75013 Paris
tèl. : + (33) 1.44.75.42.55
contact@soat.fr - www.soat.fr

Retrouvez-nous sur