Grille des critères d'évaluation code

Classe	Code	Désignation	Bien	Assez bien	Moyen	Insuffisant
7 :Code	1	Produire un code dont les « modules » qui les constituent (par ex. les classes) sont organisés de manière pertinente et qui respectent les conventions d'écriture d'un code indenté	Les « modules » (classes) constituant l'application sont organisés en paquetages opportuns et suivant une logique claire. Les conventions d'indentation du code sont celles définies par le langage ou l'outil	Les « modules » (classes) constituant l'application sont organisés en paquetages suivant une logique claire mais des paquetages sont superflus. Les conventions d'indentation du code sont celles définies par le langage ou l'outil	Les « modules » (classes) constituant l'application sont organisés en paquetages qui ne suivent pas de logique claire. Les conventions d'indentation du code sont celles définies par le langage ou l'outil	Les « modules » (classes) constituant l'application sont disponibles au sein du même paquetage
7 :Code	2	Produire une documentation de qualité d'un logiciel ou d'un « module » (par ex. classe) du logiciel en utilisant un outil (par ex. Javadoc)	Toute la documentation est produite en utilisant un outil. Elle est riche en informations et respecte les conventions du langage (balisage de l'outil de documentation). Le corps des méthodes est commenté avec des commentaires opportuns qui facilitent la compréhension de leur fonctionnement interne	Toute la documentation est produite en utilisant un outil. Elle est riche en informations et respecte les conventions du langage (balisage de l'outil de documentation). Le corps des méthodes est commenté avec des commentaires opportuns. Certaines portions complexes du code interne aux méthodes nécessitent des commentaires supplémentaires	La documentation produite en utilisant un outil est incomplète. Il manque des informations nécessaires au bon usage de l'API du logiciel ou du « module » du logiciel. Le corps des méthodes est commenté avec des commentaires opportuns. Certaines portions complexes du code interne aux méthodes nécessitent des commentaires supplémentaires	Aucune documentation n'est produite avec un outil adéquat. Aucun commentaire du corps des méthodes n'est produit ou, s'il y en a, les commentaires sont superflus ou ne facilitent pas la compréhension du fonctionnement interne des méthodes
7 :Code	3	Employer la factorisation de code « verticale » (appel des méthodes héritées des classes parentes - utilisation du mot-clé *super*) et « horizontale » (dans une même classe décomposer son code de manière à limiter la taille des méthodes et constructeurs. Le principe est d'avoir des méthodes ciblant un unique objectif, qu'elles soient réutilisables et compréhensibles – assez souvent des méthodes entre 30 et 40 lignes en moyenne)	Les factorisations « verticale » et « horizontale » sont utilisées de manière pertinente et correcte. Une erreur minime est tolérée (par ex. un oubli de factorisation ou une factorisation non pertinente)	La factorisation « verticale » est utilisée de manière pertinente et correcte. Du code de plusieurs méthodes et constructeurs pourrait être décomposé pour limiter leur taille et aucune justification n'est apportée	La factorisation « verticale » est utilisée mais elle n'est pas justifiée. Du code de plusieurs méthodes et constructeurs pourrait être décomposé pour limiter leur taille et aucune justification n'est apportée	La factorisation « verticale » n'est pas utilisée. Les méthodes et constructeurs sont souvent trop longs et complexes

7 :Code	4	Identifier les paramètres configurables de l'application et les organiser selon les conventions et les bonnes pratiques (regroupement en début de classe, constantes, *Enumeration*)	Tous les paramètres sont identifiés et organisés correctement. Les classes dédiées à la gestion des paramètres sont exploitées à bon escient	Tous les paramètres sont identifiés. Les informations sont codées dans des variables ou constantes	Certains paramètres sont identifiés. Les informations sont codées dans des variables ou constantes	Aucun paramètre n'est identifié. Les informations de paramétrage sont en dur dans le code à plusieurs endroits
7 :Code	5	Appliquer une stratégie de gestion des préconditions des méthodes qui respecte le contrat de la spécification (exception, valeur par défaut, interruption de la méthode (*return*), etc.) et l'implanter en utilisant le mécanismes des exceptions quand ceci est nécessaire (création d'exceptions propres à l'application en cas de besoin, lever, propager, capturer)	La stratégie de gestion des préconditions des méthodes respecte le contrat de spécification de celles-ci. Lorsque ceci est nécessaire, le mécanisme des exceptions est utilisé de manière pertinente et sans erreurs (les exceptions sont opportunes et explicites (nommage). Elles sont levées dans la bonne classe et sont traitées (capturées ou propagées) correctement)	La stratégie de gestion des préconditions des méthodes respecte le contrat de spécification de celles-ci. Lorsque ceci est nécessaire, le mécanisme des exceptions est utilisé avec des erreurs (des exceptions sont crées mais leur nommage ou leur traitement est inadapté (lever une exception générale, pas de capture, etc.))	La stratégie de gestion des préconditions des méthodes ne respecte pas le contrat de spécification de celles-ci. Lorsque ceci est nécessaire, le mécanisme des exceptions est utilisé mais avec des erreurs (des exceptions sont crées mais leur nommage ou leur traitement est inadapté (lever une exception générale, pas de capture, etc.))	Aucune stratégie de gestion des préconditions des méthodes n'est appliquée (la précondition des méthodes n'est pas testée)
7 :Code	6	Construire un code en corrélation avec la conception	Toutes les classes et méthodes du diagramme de classes sont présentes dans le code. Les appels de méthode sont cohérents avec les diagrammes de séquences	Toutes les classes et méthodes du diagramme de classes sont présentes dans le code. Certains appels de méthode ne sont pas cohérents avec les diagrammes de séquences	Certaines classes et méthodes du diagramme de classes sont présentes dans le code mais d'autres sont manquantes ou de nouvelles sont présentes. Aucun appel de méthode n'est cohérent avec les diagrammes de séquences	Les classes et méthodes du code n'ont aucun rapport avec les classes et méthodes des diagramme de classes et de séquences

Commentaires généraux :