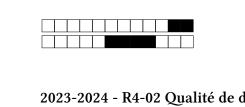
		11
0 0		Veuillez coder votre numéro d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous. Nom et prénom:
Les réponses doivent être fournies uni prise en compte dans la notation. Les ca	nique n'est autor ndique qu'il peu quement sur les f ses doivent être	-
Question 1 & Simuler des collaborate	eurs - Cochez les	assertions vraies :
programmation courants. Il n'est pas toujours possible de cré	es collaborateurs er un collaborate test permet en g inément appelé e	s simulés dans à peu près tous les langages de eur simulé "à la main" pour une classe donnée. énéral de limiter le coût d'exécution d'un test. en anglais "mock object".
Question 2 & Bénéfices et limites des	tests automatis	és (TA) - Cochez les assertions vraies :
Les TA permettent de limiter le nor Les TA permettent d'effectuer des c Les TA peuvent contenir des bugs. Les TA permettent de s'assurer qu'u	changements dan	
Question 3 . Tests unitaires vs tests	d'intégration - (Cochez les assertions vraies :
Les tests unitaires créent toujours l	es vrais collabora général plus de r en général plus ra	ateurs pour tester un objet. essources que les tests d'intégration. apidement que les tests unitaires.

Question 4 Répartition des tests - Dans une application, quelle répartition de quantités de tests est la plus pertinente : 80% de tests UI, 15% de tests d'intégration, 5% de tests unitaires. 80% de tests d'intégration, 15% de tests unitaires, 5% de tests UI. 80% de tests unitaires, 15% de tests d'intégration, 5% de tests UI. Question 5 🐥 Code coverage - Cochez les assertions vraies : Dans un projet agile, l'objectif de couverture de code par les tests est un bon critère à ajouter à la définition de fini. La couverture du code par les tests permet de savoir quelles classes, méthodes et lignes de codes sont invoquées par les tests. La couverture du code par les tests permet de mesurer objectivement la qualité des tests. En TDD, la couverture du code par les tests n'est pas nécéssairement de 100%. Question 6 🌲 TDD - Cochez les assertions vraies : En TDD, on code tous les tests d'une classe avant de coder toutes les méthodes de la classe. Le refactoring est une étape du TDD qu'un développeur doit entreprendre après qu'un test échoue. Le TDD améliore la conception d'un programme. Le refactoring consiste à remanier le code d'un programme. En TDD, dès qu'un test passe, on passe à l'écriture d'un nouveau test. Question 7 🐥 Suite de tests - Cochez les assertions vraies : Une bonne suite de tests est une suite de tests où les tests dépendent les uns des autres. Les tests des cas d'erreur suffisent pour tester la robustesse d'un programme. Un test en erreur indique en général un bug dans le test. Une bonne suite de tests teste explicitement uniquement l'interface publique de l'objet testé. Un test en échec indique en général un bug dans le programme principal.



3 3 3 3 3 3 3 3	← Veuillez coder votre numéro
	d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous.
	Nom et prénom :
Durée totale : 42 mn (56 mn pour les tiers temps) Aucun document ni dispositif électronique n'est a Le symbole "trèfle" sur une question indique qu'i Les réponses doivent être fournies uniquement su prise en compte dans la notation. Les cases doivent en compte dans la notation. Pour modifier une répo re-dessiner la case à cocher.	autorisé. Il peut être attendu plus d'une réponse correcte. Ir les feuilles d'énoncé. Aucune autre réponse ne sera têtre entièrement coloriées en noir pour être prises
Question 1 & Code coverage - Cochez les assert	ions vraies : le code par les tests est un bon critère à ajouter à la
définition de fini.	ac code par les tests est un bon effecte à ajouter à la
La couverture du code par les tests permet de 1	mesurer objectivement la qualité des tests.
La couverture du code par les tests permet de sont invoquées par les tests.	e savoir quelles classes, méthodes et lignes de codes
En TDD, la couverture du code par les tests n'e	est pas nécéssairement de 100%.
Question 2 Répartition des tests - Dans une ap la plus pertinente :	plication, quelle répartition de quantités de tests est
80% de tests unitaires, 15% de tests d'intégration	on, 5% de tests UI.
80% de tests UI, 15% de tests d'intégration, 5% de	de tests unitaires.
80% de tests d'intégration, 15% de tests unitaire	es, 5% de tests UI.
Question 3 - Coche	ez les assertions vraies :
☐ Il n'est pas possible de créer des collaborateurs	s simulés dans des tests en PHP.
Un collaborateur simulé est communément app	pelé en anglais "mock object".
Simuler des collaborateurs dans un test permet	t en général de limiter le coût d'exécution d'un test.
Il n'est pas toujours possible de créer un collab	porateur simulé "à la main" pour une classe donnée.
Il existe des librairies pour créer des collabora programmation courants.	ateurs simulés dans à peu près tous les langages de
On utilise des collaborateurs simulés en généra	al dans les tests d'intégration.



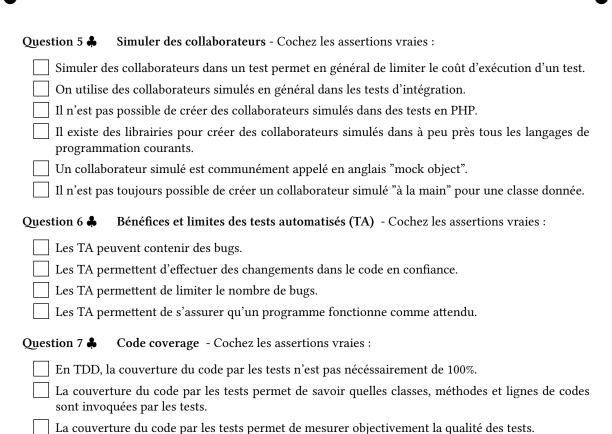
0 0	 ✓ Veuillez coder votre numéro d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous. Nom et prénom :
Durée totale : 42 mn (56 mn pour les tiers temps). Pa Aucun document ni dispositif électronique n'est auto Le symbole "trèfle" sur une question indique qu'il pe Les réponses doivent être fournies uniquement sur le prise en compte dans la notation. Les cases doivent être en compte dans la notation. Pour modifier une réponse re-dessiner la case à cocher.	orisé. eut être attendu plus d'une réponse correcte. es feuilles d'énoncé. Aucune autre réponse ne sera re entièrement coloriées en noir pour être prises
Question 1 ♣ Bénéfices et limites des tests automat Les TA permettent de limiter le nombre de bugs. Les TA permettent d'effectuer des changements de les TA peuvent contenir des bugs. Les TA permettent de s'assurer qu'un programme	
Question 2 ♣ Suite de tests - Cochez les assertions v Les tests des cas d'erreur suffisent pour tester la re Un test en échec indique en général un bug dans l Un test en erreur indique en général un bug dans l Une bonne suite de tests teste explicitement uniqu Une bonne suite de tests est une suite de tests où l	obustesse d'un programme. le programme principal. le test. uement l'interface publique de l'objet testé.
Question 3 & Code coverage - Cochez les assertions	s vraies :
La couverture du code par les tests permet de sar sont invoquées par les tests.	
La couverture du code par les tests permet de mes Dans un projet agile, l'objectif de couverture de c définition de fini.	
En TDD, la couverture du code par les tests n'est p	pas nécéssairement de 100%.

Question 4 A TDD - Cochez les assertions vraies :
Le TDD améliore la conception d'un programme. Le refactoring est une étape du TDD qu'un développeur doit entreprendre après qu'un test échoue Le refactoring consiste à remanier le code d'un programme. En TDD, dès qu'un test passe, on passe à l'écriture d'un nouveau test. En TDD, on code tous les tests d'une classe avant de coder toutes les méthodes de la classe.
Question 5 - Simuler des collaborateurs - Cochez les assertions vraies :
 Un collaborateur simulé est communément appelé en anglais "mock object". Il n'est pas toujours possible de créer un collaborateur simulé "à la main" pour une classe donnée. Il n'est pas possible de créer des collaborateurs simulés dans des tests en PHP. On utilise des collaborateurs simulés en général dans les tests d'intégration. Il existe des librairies pour créer des collaborateurs simulés dans à peu près tous les langages d programmation courants. Simuler des collaborateurs dans un test permet en général de limiter le coût d'exécution d'un test.
Question 6 Répartition des tests - Dans une application, quelle répartition de quantités de tests es la plus pertinente :
80% de tests UI, 15% de tests d'intégration, 5% de tests unitaires. 80% de tests unitaires, 15% de tests d'intégration, 5% de tests UI. 80% de tests d'intégration, 15% de tests unitaires, 5% de tests UI.
Question 7 - Cochez les assertions vraies :
Les tests unitaires consomment en général plus de ressources que les tests d'intégration. Les tests unitaires créent toujours les vrais collaborateurs pour tester un objet. Les tests d'intégration vérifient le fonctionnement de "vrais" objets en interactions. Les tests d'intégration s'exécutent en général plus rapidement que les tests unitaires.

2023-2024 - R4-02 Qualité de développement - CC1 $\Box_1 \Box_1 \Box$ $\Box_1 \Box_1 \Box_1$ 2 2 2 Veuillez coder votre numéro 3 3 3 | 3 d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous. 5 5 Nom et prénom : 9 9 Durée totale : 42 mn (56 mn pour les tiers temps). Partie QCM sur 7 points. Aucun document ni dispositif électronique n'est autorisé. Le symbole "trèfle" sur une question indique qu'il peut être attendu plus d'une réponse correcte. Les réponses doivent être fournies uniquement sur les feuilles d'énoncé. Aucune autre réponse ne sera prise en compte dans la notation. Les cases doivent être entièrement coloriées en noir pour être prises en compte dans la notation. Pour modifier une réponse, vous pouvez utiliser un correcteur blanc, sans re-dessiner la case à cocher. Ouestion 1 🌲 Tests unitaires vs tests d'intégration - Cochez les assertions vraies : Les tests unitaires créent toujours les vrais collaborateurs pour tester un objet. Les tests d'intégration vérifient le fonctionnement de "vrais" objets en interactions. Les tests d'intégration s'exécutent en général plus rapidement que les tests unitaires. Les tests unitaires consomment en général plus de ressources que les tests d'intégration. Question 2 🚓 Suite de tests - Cochez les assertions vraies : Un test en échec indique en général un bug dans le programme principal. Une bonne suite de tests est une suite de tests où les tests dépendent les uns des autres. Les tests des cas d'erreur suffisent pour tester la robustesse d'un programme. Un test en erreur indique en général un bug dans le test. Une bonne suite de tests teste explicitement uniquement l'interface publique de l'objet testé. Répartition des tests - Dans une application, quelle répartition de quantités de tests est Question 3 la plus pertinente : 80% de tests UI, 15% de tests d'intégration, 5% de tests unitaires. 80% de tests d'intégration, 15% de tests unitaires, 5% de tests UI. 80% de tests unitaires, 15% de tests d'intégration, 5% de tests UI. Ouestion 4 🌲 TDD - Cochez les assertions vraies : Le refactoring est une étape du TDD qu'un développeur doit entreprendre après qu'un test échoue. Le TDD améliore la conception d'un programme. En TDD, dès qu'un test passe, on passe à l'écriture d'un nouveau test.

En TDD, on code tous les tests d'une classe avant de coder toutes les méthodes de la classe.

Le refactoring consiste à remanier le code d'un programme.

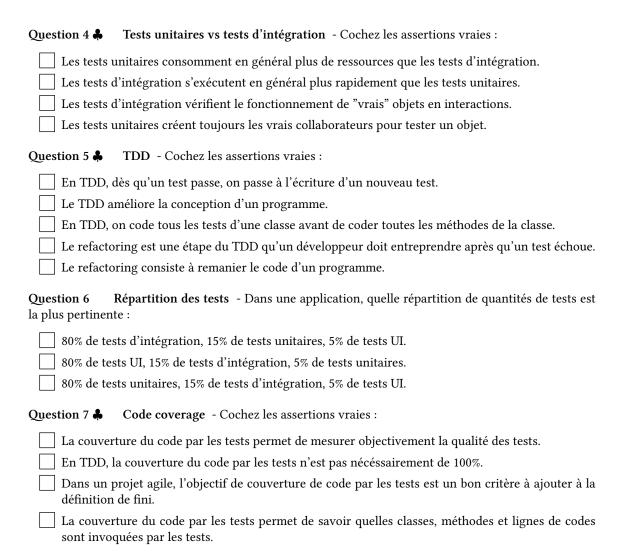


Dans un projet agile, l'objectif de couverture de code par les tests est un bon critère à ajouter à la

définition de fini.



0 0	
Durée totale : 42 mn (56 mn pour les tiers temps). Pa Aucun document ni dispositif électronique n'est auto Le symbole "trèfle" sur une question indique qu'il pe Les réponses doivent être fournies uniquement sur le prise en compte dans la notation. Les cases doivent être en compte dans la notation. Pour modifier une réponse re-dessiner la case à cocher.	orisé. eut être attendu plus d'une réponse correcte. es feuilles d'énoncé. Aucune autre réponse ne sera re entièrement coloriées en noir pour être prises
Question 1 ♣ Suite de tests - Cochez les assertions v Les tests des cas d'erreur suffisent pour tester la re Un test en échec indique en général un bug dans l Un test en erreur indique en général un bug dans Une bonne suite de tests teste explicitement uniqu Une bonne suite de tests est une suite de tests où l	obustesse d'un programme. e programme principal. le test. uement l'interface publique de l'objet testé.
Question 2 ♣ Bénéfices et limites des tests automat Les TA permettent d'effectuer des changements de Les TA peuvent contenir des bugs. Les TA permettent de s'assurer qu'un programme Les TA permettent de limiter le nombre de bugs.	
On utilise des collaborateurs - Cochez le ☐ On utilise des collaborateurs simulés en général d ☐ Un collaborateur simulé est communément appele ☐ Simuler des collaborateurs dans un test permet en ☐ Il n'est pas possible de créer des collaborateurs sir ☐ Il existe des librairies pour créer des collaborateur programmation courants.	ans les tests d'intégration. é en anglais "mock object". n général de limiter le coût d'exécution d'un test. nulés dans des tests en PHP.
☐ Il n'est pas toujours possible de créer un collabora	teur simulé "à la main" pour une classe donnée.





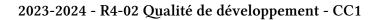
0 0	
Durée totale : 42 mn (56 mn pour les tiers temps). Pa Aucun document ni dispositif électronique n'est aut Le symbole "trèfle" sur une question indique qu'il pe Les réponses doivent être fournies uniquement sur le prise en compte dans la notation. Les cases doivent être en compte dans la notation. Pour modifier une répons re-dessiner la case à cocher.	orisé. eut être attendu plus d'une réponse correcte. es feuilles d'énoncé. Aucune autre réponse ne sera re entièrement coloriées en noir pour être prises
Ouestion 1 ♣ Simuler des collaborateurs - Cochez l Simuler des collaborateurs dans un test permet er Il n'est pas toujours possible de créer un collaborateur Il existe des librairies pour créer des collaborateur programmation courants. Il n'est pas possible de créer des collaborateurs sin Un collaborateur simulé est communément appele On utilise des collaborateurs simulés en général des	a général de limiter le coût d'exécution d'un test. ateur simulé "à la main" pour une classe donnée. urs simulés dans à peu près tous les langages de mulés dans des tests en PHP. é en anglais "mock object".
Question 2 Répartition des tests - Dans une applie la plus pertinente :	cation, quelle répartition de quantités de tests est
80% de tests UI, 15% de tests d'intégration, 5% de tests d'intégration, 5% de tests d'intégration, 5% de tests d'intégration, 5% de tests unitaires, 5% de tests	5% de tests UI.
Question 3 ♣ TDD - Cochez les assertions vraies :	
Le refactoring est une étape du TDD qu'un dévelor Le refactoring consiste à remanier le code d'un pr En TDD, on code tous les tests d'une classe avant Le TDD améliore la conception d'un programme. En TDD, dès qu'un test passe, on passe à l'écriture.	ogramme. de coder toutes les méthodes de la classe.



Ç	uestion 4 - Suite de tests - Cochez les assertions vraies :
	Une bonne suite de tests teste explicitement uniquement l'interface publique de l'objet testé.
	Une bonne suite de tests est une suite de tests où les tests dépendent les uns des autres.
	Un test en échec indique en général un bug dans le programme principal.
	Un test en erreur indique en général un bug dans le test.
	Les tests des cas d'erreur suffisent pour tester la robustesse d'un programme.
Ç	uestion 5 ♣ Code coverage - Cochez les assertions vraies :
	La couverture du code par les tests permet de savoir quelles classes, méthodes et lignes de codes sont invoquées par les tests.
	Dans un projet agile, l'objectif de couverture de code par les tests est un bon critère à ajouter à la définition de fini.
	En TDD, la couverture du code par les tests n'est pas nécéssairement de 100%.
	La couverture du code par les tests permet de mesurer objectivement la qualité des tests.
Ç	uestion 6 ♣ Bénéfices et limites des tests automatisés (TA) - Cochez les assertions vraies :
	Les TA permettent de limiter le nombre de bugs.
	Les TA peuvent contenir des bugs.
	Les TA permettent de s'assurer qu'un programme fonctionne comme attendu.
	Les TA permettent d'effectuer des changements dans le code en confiance.
Ç	uestion 7 ♣ Tests unitaires vs tests d'intégration - Cochez les assertions vraies :
	Les tests unitaires créent toujours les vrais collaborateurs pour tester un objet.
	Les tests d'intégration s'exécutent en général plus rapidement que les tests unitaires.
	Les tests unitaires consomment en général plus de ressources que les tests d'intégration.
	Les tests d'intégration vérifient le fonctionnement de "vrais" objets en interactions.

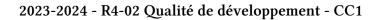
\square 3	← Veuillez coder votre numéro
\square_4 \square_4 \square_4 \square_4 \square_4 \square_4 \square_4	d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous.
	Nom et prénom :
Durée totale : 42 mn (56 mn pour les tiers temps). Aucun document ni dispositif électronique n'est au Le symbole "trèfle" sur une question indique qu'il p Les réponses doivent être fournies uniquement sur prise en compte dans la notation. Les cases doivent être compte dans la notation. Pour modifier une réponre-dessiner la case à cocher.	utorisé. peut être attendu plus d'une réponse correcte. les feuilles d'énoncé. Aucune autre réponse ne sera être entièrement coloriées en noir pour être prises
Un collaborateur simulé est communément appe Il existe des librairies pour créer des collaborate programmation courants.	rateur simulé "à la main" pour une classe donnée. elé en anglais "mock object". eurs simulés dans à peu près tous les langages de en général de limiter le coût d'exécution d'un test.
On utilise des collaborateurs simulés en général	dans les tests d'intégration.
Question 2 🌲 Code coverage - Cochez les assertio	ns vraies :
En TDD, la couverture du code par les tests n'es	t pas nécéssairement de 100%.
La couverture du code par les tests permet de me	esurer objectivement la qualité des tests.
La couverture du code par les tests permet de s sont invoquées par les tests.	savoir quelles classes, méthodes et lignes de codes
Dans un projet agile, l'objectif de couverture de définition de fini.	code par les tests est un bon critère à ajouter à la
Question 3 4 Tests unitaires vs tests d'intégration	1 - Cochez les assertions vraies :
Les tests unitaires créent toujours les vrais colla	borateurs pour tester un objet.
Les tests d'intégration s'exécutent en général plu	us rapidement que les tests unitaires.
Les tests unitaires consomment en général plus	de ressources que les tests d'intégration.
Les tests d'intégration vérifient le fonctionneme	nt de "vrais" objets en interactions.

Question 4 - TDD - Cochez les assertions vraies :
En TDD, on code tous les tests d'une classe avant de coder toutes les méthodes de la classe.
Le TDD améliore la conception d'un programme.
Le refactoring consiste à remanier le code d'un programme.
En TDD, dès qu'un test passe, on passe à l'écriture d'un nouveau test.
Le refactoring est une étape du TDD qu'un développeur doit entreprendre après qu'un test échoue
Question 5 Répartition des tests - Dans une application, quelle répartition de quantités de tests es la plus pertinente :
80% de tests UI, 15% de tests d'intégration, 5% de tests unitaires.
80% de tests d'intégration, 15% de tests unitaires, 5% de tests UI.
80% de tests unitaires, 15% de tests d'intégration, 5% de tests UI.
Question 6 & Bénéfices et limites des tests automatisés (TA) - Cochez les assertions vraies :
Les TA peuvent contenir des bugs.
Les TA permettent de limiter le nombre de bugs.
Les TA permettent de s'assurer qu'un programme fonctionne comme attendu.
Les TA permettent d'effectuer des changements dans le code en confiance.
Question 7 - Suite de tests - Cochez les assertions vraies :
Les tests des cas d'erreur suffisent pour tester la robustesse d'un programme.
Une bonne suite de tests est une suite de tests où les tests dépendent les uns des autres.
Un test en erreur indique en général un bug dans le test.
Une bonne suite de tests teste explicitement uniquement l'interface publique de l'objet testé.
Un test en échec indique en général un bug dans le programme principal.

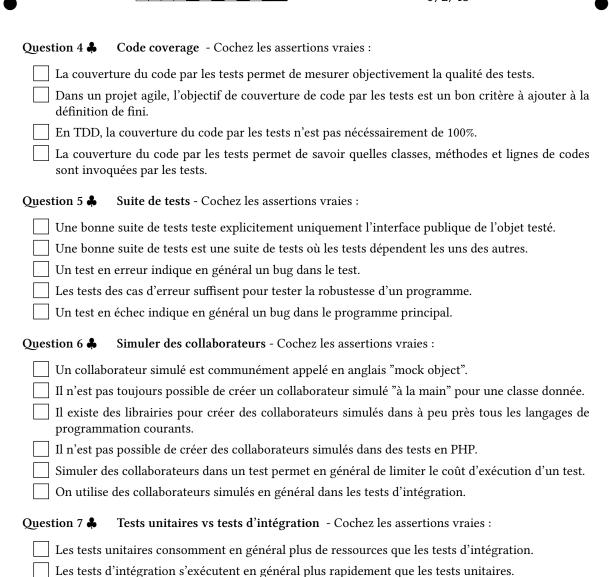


\square_2 \square_2 \square_2 \square_2 \square_2 \square_2 \square_2	
	← Veuillez coder votre numéro
\square_4 \square_4 \square_4 \square_4 \square_4 \square_4 \square_4	d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous.
	dalis la case ci-dessous.
	Nom et prénom :
Durée totale : 42 mn (56 mn pour les tiers temps). I Aucun document ni dispositif électronique n'est au Le symbole "trèfle" sur une question indique qu'il p Les réponses doivent être fournies uniquement sur l prise en compte dans la notation. Les cases doivent ê en compte dans la notation. Pour modifier une répon re-dessiner la case à cocher.	torisé. Deut être attendu plus d'une réponse correcte. les feuilles d'énoncé. Aucune autre réponse ne sera tre entièrement coloriées en noir pour être prises
Question 1 - Suite de tests - Cochez les assertions	vraies :
Un test en erreur indique en général un bug dans	s le test.
Une bonne suite de tests est une suite de tests où	
Une bonne suite de tests teste explicitement unic	
Les tests des cas d'erreur suffisent pour tester la	
Un test en échec indique en général un bug dans	
Question 2 - Bénéfices et limites des tests automa	atisés (TA) - Cochez les assertions vraies :
Les TA peuvent contenir des bugs.	
Les TA permettent d'effectuer des changements o	dans le code en confiance.
Les TA permettent de s'assurer qu'un programm	e fonctionne comme attendu.
Les TA permettent de limiter le nombre de bugs.	
Question 3 & Code coverage - Cochez les assertion	ns vraies :
En TDD, la couverture du code par les tests n'est	pas nécéssairement de 100%.
Dans un projet agile, l'objectif de couverture de définition de fini.	code par les tests est un bon critère à ajouter à la
La couverture du code par les tests permet de sa sont invoquées par les tests.	avoir quelles classes, méthodes et lignes de codes
La couverture du code par les tests permet de me	esurer objectivement la qualité des tests.

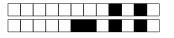
Question 4 - Cochez les assertions vraies :
Les tests unitaires créent toujours les vrais collaborateurs pour tester un objet.
Les tests d'intégration vérifient le fonctionnement de "vrais" objets en interactions.
Les tests d'intégration s'exécutent en général plus rapidement que les tests unitaires.
Les tests unitaires consomment en général plus de ressources que les tests d'intégration.
Question 5 - Simuler des collaborateurs - Cochez les assertions vraies :
Il existe des librairies pour créer des collaborateurs simulés dans à peu près tous les langages de programmation courants.
On utilise des collaborateurs simulés en général dans les tests d'intégration.
Il n'est pas toujours possible de créer un collaborateur simulé "à la main" pour une classe donnée.
Simuler des collaborateurs dans un test permet en général de limiter le coût d'exécution d'un test.
Un collaborateur simulé est communément appelé en anglais "mock object".
Il n'est pas possible de créer des collaborateurs simulés dans des tests en PHP.
Question 6 ♣ TDD - Cochez les assertions vraies :
En TDD, dès qu'un test passe, on passe à l'écriture d'un nouveau test.
Le refactoring est une étape du TDD qu'un développeur doit entreprendre après qu'un test échoue
Le TDD améliore la conception d'un programme.
Le refactoring consiste à remanier le code d'un programme.
En TDD, on code tous les tests d'une classe avant de coder toutes les méthodes de la classe.
Question 7 Répartition des tests - Dans une application, quelle répartition de quantités de tests es la plus pertinente :
80% de tests unitaires, 15% de tests d'intégration, 5% de tests UI.
80% de tests UI, 15% de tests d'intégration, 5% de tests unitaires.
80% de tests d'intégration 15% de tests unitaires 5% de tests III



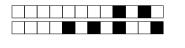
\square 2	
\square_3 \square_3 \square_3 \square_3 \square_3 \square_3 \square_3	← Veuillez coder votre numéro
	d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous.
	dans la case ci-dessous.
555555	Nom et prénom :
8 8 8 8 8 8 8 8	
9 9 9 9 9 9	
Durée totale : 42 mn (56 mn pour les tiers temps). Pa Aucun document ni dispositif électronique n'est auto Le symbole "trèfle" sur une question indique qu'il pe Les réponses doivent être fournies uniquement sur le prise en compte dans la notation. Les cases doivent être en compte dans la notation. Pour modifier une réponse	orisé. eut être attendu plus d'une réponse correcte. es feuilles d'énoncé. Aucune autre réponse ne sera re entièrement coloriées en noir pour être prises
re-dessiner la case à cocher.	
Question 1 & Bénéfices et limites des tests automat	isés (TA) - Cochez les assertions vraies :
Les TA permettent de s'assurer qu'un programme	
	fonctionne comme attendu.
Les TA peuvent contenir des bugs.	
Les TA permettent de limiter le nombre de bugs.	
Les TA permettent d'effectuer des changements d	ans le code en confiance.
Question 2 ♣ TDD - Cochez les assertions vraies :	
En TDD, on code tous les tests d'une classe avant	de coder toutes les méthodes de la classe.
En TDD, dès qu'un test passe, on passe à l'écriture	e d'un nouveau test.
Le refactoring est une étape du TDD qu'un dévelo	
Le refactoring consiste à remanier le code d'un pr	
Le TDD améliore la conception d'un programme.	-8
Le 122 amenore la conception a un programme.	
Question 3 Répartition des tests - Dans une applie la plus pertinente :	eation, quelle répartition de quantités de tests est
80% de tests unitaires, 15% de tests d'intégration,	5% de tests UI.
80% de tests d'intégration, 15% de tests unitaires,	
80% de tests UI, 15% de tests d'intégration, 5% de t	
22.0000 ex, 10% de teste à missilation, 5% de t	



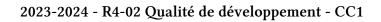
Les tests unitaires créent toujours les vrais collaborateurs pour tester un objet. Les tests d'intégration vérifient le fonctionnement de "vrais" objets en interactions.



\square_3 \square_3 \square_3 \square_3 \square_3 \square_3 \square_3	← Veuillez coder votre numéro
	d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous.
	dans la case el dessous.
	Nom et prénom :
Durée totale : 42 mn (56 mn pour les tiers temps) Aucun document ni dispositif électronique n'est a Le symbole "trèfle" sur une question indique qu'i Les réponses doivent être fournies uniquement su prise en compte dans la notation. Les cases doivent en compte dans la notation. Pour modifier une répo re-dessiner la case à cocher.	autorisé. l peut être attendu plus d'une réponse correcte. ur les feuilles d'énoncé. Aucune autre réponse ne sera cêtre entièrement coloriées en noir pour être prises
Question 1 Répartition des tests - Dans une ap la plus pertinente :	plication, quelle répartition de quantités de tests est
	1
80% de tests UI, 15% de tests d'intégration, 5% d	
80% de tests d'intégration, 15% de tests unitaire	
80% de tests unitaires, 15% de tests d'intégration	on, 5% de tests UI.
Question 2 🌲 Tests unitaires vs tests d'intégration	on - Cochez les assertions vraies :
Les tests d'intégration s'exécutent en général p	olus rapidement que les tests unitaires.
Les tests unitaires créent toujours les vrais col	laborateurs pour tester un objet.
Les tests unitaires consomment en général plu	s de ressources que les tests d'intégration.
Les tests d'intégration vérifient le fonctionnem	nent de "vrais" objets en interactions.
Question 3 & Simuler des collaborateurs - Coche	ez les assertions vraies :
Il n'est pas toujours possible de créer un collab	orateur simulé "à la main" pour une classe donnée.
Il existe des librairies pour créer des collabora programmation courants.	ateurs simulés dans à peu près tous les langages de
Simuler des collaborateurs dans un test perme	t en général de limiter le coût d'exécution d'un test.
☐ Il n'est pas possible de créer des collaborateurs	aineulás dans das tasts on DIID
	simules dans des tests en PHP.
Un collaborateur simulé est communément ap	



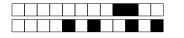
Question 4 🤻	Suite de tests - Cochez les assertions vraies :
Un test	en échec indique en général un bug dans le programme principal.
Les test	s des cas d'erreur suffisent pour tester la robustesse d'un programme.
Un test	en erreur indique en général un bug dans le test.
Une bo	nne suite de tests est une suite de tests où les tests dépendent les uns des autres.
Une bo	nne suite de tests teste explicitement uniquement l'interface publique de l'objet testé.
Question 5 🧍	Code coverage - Cochez les assertions vraies :
	n projet agile, l'objectif de couverture de code par les tests est un bon critère à ajouter à la on de fini.
	verture du code par les tests permet de savoir quelles classes, méthodes et lignes de codes voquées par les tests.
En TDI), la couverture du code par les tests n'est pas nécéssairement de 100%.
La couv	verture du code par les tests permet de mesurer objectivement la qualité des tests.
Question 6 🧍	TDD - Cochez les assertions vraies :
En TDI	D, dès qu'un test passe, on passe à l'écriture d'un nouveau test.
Le TDI	Daméliore la conception d'un programme.
En TDI	D, on code tous les tests d'une classe avant de coder toutes les méthodes de la classe.
Le refa	ctoring est une étape du TDD qu'un développeur doit entreprendre après qu'un test échoue.
Le refa	ctoring consiste à remanier le code d'un programme.
Question 7 🧍	Bénéfices et limites des tests automatisés (TA) - Cochez les assertions vraies :
Les TA	permettent de s'assurer qu'un programme fonctionne comme attendu.
Les TA	permettent d'effectuer des changements dans le code en confiance.
Les TA	permettent de limiter le nombre de bugs.
Les TA	neuvent contenir des bugs.



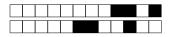
0 0	← Veuillez coder votre numéro d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous. Nom et prénom:
Durée totale : 42 mn (56 mn pour les tiers temps). Pa Aucun document ni dispositif électronique n'est aut Le symbole "trèfle" sur une question indique qu'il pa Les réponses doivent être fournies uniquement sur le prise en compte dans la notation. Les cases doivent être en compte dans la notation. Pour modifier une répons re-dessiner la case à cocher.	orisé. eut être attendu plus d'une réponse correcte. es feuilles d'énoncé. Aucune autre réponse ne sera re entièrement coloriées en noir pour être prises
Question 1 ♣ Simuler des collaborateurs - Cochez l Il n'est pas possible de créer des collaborateurs sin Il n'est pas toujours possible de créer un collaborateurs dans un test permet en Un collaborateur simulé est communément appele On utilise des collaborateurs simulés en général d Il existe des librairies pour créer des collaborateur programmation courants.	mulés dans des tests en PHP. ateur simulé "à la main" pour une classe donnée. a général de limiter le coût d'exécution d'un test. é en anglais "mock object". ans les tests d'intégration.
Question 2 ♣ Tests unitaires vs tests d'intégration Les tests d'intégration vérifient le fonctionnement Les tests unitaires consomment en général plus de Les tests d'intégration s'exécutent en général plus Les tests unitaires créent toujours les vrais collabe	t de "vrais" objets en interactions. e ressources que les tests d'intégration. s rapidement que les tests unitaires.
Question 3 ♣ TDD - Cochez les assertions vraies : Le TDD améliore la conception d'un programme. Le refactoring est une étape du TDD qu'un dévelo En TDD, dès qu'un test passe, on passe à l'écriture En TDD, on code tous les tests d'une classe avant Le refactoring consiste à remanier le code d'un pr	e d'un nouveau test. de coder toutes les méthodes de la classe.

Question 4	Code coverage - Cochez les assertions vraies :
	verture du code par les tests permet de savoir quelles classes, méthodes et lignes de codes voquées par les tests.
	n projet agile, l'objectif de couverture de code par les tests est un bon critère à ajouter à la on de fini.
La cou	verture du code par les tests permet de mesurer objectivement la qualité des tests.
En TDl	O, la couverture du code par les tests n'est pas nécéssairement de 100%.
Question 5 🎝	Bénéfices et limites des tests automatisés (TA) - Cochez les assertions vraies :
Les TA	permettent de limiter le nombre de bugs.
Les TA	permettent d'effectuer des changements dans le code en confiance.
Les TA	permettent de s'assurer qu'un programme fonctionne comme attendu.
Les TA	peuvent contenir des bugs.
Question 6	Suite de tests - Cochez les assertions vraies :
Un test	en échec indique en général un bug dans le programme principal.
Une bo	nne suite de tests est une suite de tests où les tests dépendent les uns des autres.
Un test	en erreur indique en général un bug dans le test.
Une bo	nne suite de tests teste explicitement uniquement l'interface publique de l'objet testé.
Les tes	ts des cas d'erreur suffisent pour tester la robustesse d'un programme.
Question 7 la plus pertin	Répartition des tests - Dans une application, quelle répartition de quantités de tests est ente :
80% de	tests d'intégration, 15% de tests unitaires, 5% de tests UI.
80% de	tests unitaires, 15% de tests d'intégration, 5% de tests UI.
80% de	tests III 15% de tests d'intégration 5% de tests unitaires

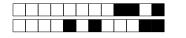
0 0	✓ Veuillez coder votre numéro d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous. Nom et prénom :
888888 99	
Durée totale : 42 mn (56 mn pour les tiers temps). Pa Aucun document ni dispositif électronique n'est auto Le symbole "trèfle" sur une question indique qu'il pe Les réponses doivent être fournies uniquement sur le prise en compte dans la notation. Les cases doivent être en compte dans la notation. Pour modifier une réponse re-dessiner la case à cocher.	orisé. out être attendu plus d'une réponse correcte. s feuilles d'énoncé. Aucune autre réponse ne sera re entièrement coloriées en noir pour être prises
Question 1 ♣ Bénéfices et limites des tests automat Les TA permettent d'effectuer des changements de Les TA permettent de limiter le nombre de bugs. Les TA permettent de s'assurer qu'un programme Les TA peuvent contenir des bugs.	
Question 2 & Code coverage - Cochez les assertions	s vraies :
Dans un projet agile, l'objectif de couverture de c définition de fini.	ode par les tests est un bon critère à ajouter à la
La couverture du code par les tests permet de sav sont invoquées par les tests.	voir quelles classes, méthodes et lignes de codes
En TDD, la couverture du code par les tests n'est p	
La couverture du code par les tests permet de mes	urer objectivement la qualité des tests.
Question 3 - Simuler des collaborateurs - Cochez le	es assertions vraies :
Simuler des collaborateurs dans un test permet en	
On utilise des collaborateurs simulés en général de	-
Il existe des librairies pour créer des collaborateu programmation courants.	irs simules dans a peu pres tous les langages de
☐ Il n'est pas possible de créer des collaborateurs sin	nulés dans des tests en PHP.
Il n'est pas toujours possible de créer un collabora	teur simulé "à la main" pour une classe donnée.
Un collaborateur simulé est communément appelé	en anglais "mock object".



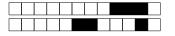
Question 4 - Suite de tests - Cochez les assertions vraies :	
Un test en échec indique en général un bug dans le programme principal.	
Les tests des cas d'erreur suffisent pour tester la robustesse d'un programme.	
Un test en erreur indique en général un bug dans le test.	
Une bonne suite de tests teste explicitement uniquement l'interface publique de l'objet testé.	
Une bonne suite de tests est une suite de tests où les tests dépendent les uns des autres.	
Question 5 A TDD - Cochez les assertions vraies :	
En TDD, on code tous les tests d'une classe avant de coder toutes les méthodes de la classe.	
Le refactoring consiste à remanier le code d'un programme.	
Le refactoring est une étape du TDD qu'un développeur doit entreprendre après qu'un test échoue.	
En TDD, dès qu'un test passe, on passe à l'écriture d'un nouveau test.	
Le TDD améliore la conception d'un programme.	
Question 6 Répartition des tests - Dans une application, quelle répartition de quantités de tests est la plus pertinente :	
80% de tests d'intégration, 15% de tests unitaires, 5% de tests UI.	
80% de tests unitaires, 15% de tests d'intégration, 5% de tests UI.	
80% de tests UI, 15% de tests d'intégration, 5% de tests unitaires.	
Question 7 ♣ Tests unitaires vs tests d'intégration - Cochez les assertions vraies :	
Les tests unitaires consomment en général plus de ressources que les tests d'intégration.	
Les tests unitaires créent toujours les vrais collaborateurs pour tester un objet.	
Les tests d'intégration vérifient le fonctionnement de "vrais" objets en interactions.	
Les tests d'intégration s'exécutent en général plus rapidement que les tests unitaires	



0 0	
Durée totale : 42 mn (56 mn pour les tiers temps). Pa Aucun document ni dispositif électronique n'est auto Le symbole "trèfle" sur une question indique qu'il pe Les réponses doivent être fournies uniquement sur le prise en compte dans la notation. Les cases doivent être en compte dans la notation. Pour modifier une réponse re-dessiner la case à cocher.	orisé. out être attendu plus d'une réponse correcte. s feuilles d'énoncé. Aucune autre réponse ne sera re entièrement coloriées en noir pour être prises
Question 1 ♣ Bénéfices et limites des tests automat Les TA permettent d'effectuer des changements de Les TA permettent de s'assurer qu'un programme Les TA peuvent contenir des bugs. Les TA permettent de limiter le nombre de bugs.	
 Question 2 ♣ Code coverage - Cochez les assertions □ La couverture du code par les tests permet de mes □ La couverture du code par les tests permet de sar sont invoquées par les tests. □ Dans un projet agile, l'objectif de couverture de c définition de fini. □ En TDD, la couverture du code par les tests n'est par les t	urer objectivement la qualité des tests. voir quelles classes, méthodes et lignes de codes ode par les tests est un bon critère à ajouter à la
Question 3 ♣ Suite de tests - Cochez les assertions v Un test en échec indique en général un bug dans l Un test en erreur indique en général un bug dans l Les tests des cas d'erreur suffisent pour tester la re Une bonne suite de tests teste explicitement uniqu Une bonne suite de tests est une suite de tests où l	e programme principal. le test. obustesse d'un programme. nement l'interface publique de l'objet testé.



Question 4 👫 TD	D - Cochez les assertions vraies :
Le refactoring	consiste à remanier le code d'un programme.
En TDD, on co	de tous les tests d'une classe avant de coder toutes les méthodes de la classe.
Le refactoring	est une étape du TDD qu'un développeur doit entreprendre après qu'un test échoue.
Le TDD amélio	ore la conception d'un programme.
En TDD, dès q	u'un test passe, on passe à l'écriture d'un nouveau test.
Question 5 Répa la plus pertinente :	rtition des tests - Dans une application, quelle répartition de quantités de tests est
80% de tests ur	nitaires, 15% de tests d'intégration, 5% de tests UI.
80% de tests Ul	I, 15% de tests d'intégration, 5% de tests unitaires.
80% de tests d'	intégration, 15% de tests unitaires, 5% de tests UI.
Question 6 & Sim	nuler des collaborateurs - Cochez les assertions vraies :
Simuler des co	llaborateurs dans un test permet en général de limiter le coût d'exécution d'un test.
Il n'est pas tou	jours possible de créer un collaborateur simulé "à la main" pour une classe donnée.
Il existe des lil	brairies pour créer des collaborateurs simulés dans à peu près tous les langages de n courants.
Un collaborate	ur simulé est communément appelé en anglais "mock object".
Il n'est pas pos	sible de créer des collaborateurs simulés dans des tests en PHP.
On utilise des	collaborateurs simulés en général dans les tests d'intégration.
Question 7 4 Tes	ets unitaires vs tests d'intégration - Cochez les assertions vraies :
Les tests d'inté	gration s'exécutent en général plus rapidement que les tests unitaires.
Les tests unita	ires créent toujours les vrais collaborateurs pour tester un objet.
Les tests d'inté	gration vérifient le fonctionnement de "vrais" objets en interactions.
Les tests unita	ires consomment en général plus de ressources que les tests d'intégration.



\square 2	
\square_3 \square_3 \square_3 \square_3 \square_3 \square_3 \square_3	← Veuillez coder votre numéro
	d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous.
	dalls la case ci-dessous.
	Nom et prénom :
9999999	
Durée totale : 42 mn (56 mn pour les tiers temps). I Aucun document ni dispositif électronique n'est au Le symbole "trèfle" sur une question indique qu'il p Les réponses doivent être fournies uniquement sur l prise en compte dans la notation. Les cases doivent êt en compte dans la notation. Pour modifier une répons re-dessiner la case à cocher.	torisé. Jeut être attendu plus d'une réponse correcte. es feuilles d'énoncé. Aucune autre réponse ne sera tre entièrement coloriées en noir pour être prises
Question 1 🌲 Tests unitaires vs tests d'intégration	- Cochez les assertions vraies :
Les tests unitaires consomment en général plus d	le ressources que les tests d'intégration.
Les tests unitaires créent toujours les vrais collab	
Les tests d'intégration s'exécutent en général plu	
Les tests d'intégration vérifient le fonctionnemer	
Question 2 . TDD - Cochez les assertions vraies :	,
En TDD, on code tous les tests d'une classe avant	
	oppeur doit entreprendre après qu'un test échoue.
En TDD, dès qu'un test passe, on passe à l'écritur	re d'un nouveau test.
Le TDD améliore la conception d'un programme	
Le refactoring consiste à remanier le code d'un p	rogramme.
Question 3 - Suite de tests - Cochez les assertions	vraies :
Un test en échec indique en général un bug dans	le programme principal.
Un test en erreur indique en général un bug dans	s le test.
Une bonne suite de tests est une suite de tests où	les tests dépendent les uns des autres.
Les tests des cas d'erreur suffisent pour tester la	robustesse d'un programme.
Une bonne suite de tests teste explicitement uniq	quement l'interface publique de l'objet testé.

Question 4 🌲 🔻 B	énéfices et limites des tests automatisés (TA) - Cochez les assertions vraies :
Les TA perm	ettent de limiter le nombre de bugs.
Les TA peuv	ent contenir des bugs.
Les TA perm	ettent d'effectuer des changements dans le code en confiance.
Les TA perm	ettent de s'assurer qu'un programme fonctionne comme attendu.
Question 5 🌲 🛚 C	ode coverage - Cochez les assertions vraies :
En TDD, la c	ouverture du code par les tests n'est pas nécéssairement de 100%.
Dans un prodéfinition de	jet agile, l'objectif de couverture de code par les tests est un bon critère à ajouter à la fini.
	re du code par les tests permet de savoir quelles classes, méthodes et lignes de codes des par les tests.
La couvertur	e du code par les tests permet de mesurer objectivement la qualité des tests.
Question 6 👫 Si	imuler des collaborateurs - Cochez les assertions vraies :
Un collabora	teur simulé est communément appelé en anglais "mock object".
Simuler des d	collaborateurs dans un test permet en général de limiter le coût d'exécution d'un test.
Il n'est pas p	ossible de créer des collaborateurs simulés dans des tests en PHP.
On utilise de	s collaborateurs simulés en général dans les tests d'intégration.
Il n'est pas to	oujours possible de créer un collaborateur simulé "à la main" pour une classe donnée.
Il existe des programmat	librairies pour créer des collaborateurs simulés dans à peu près tous les langages de ion courants.
Question 7 Réj la plus pertinente :	partition des tests - Dans une application, quelle répartition de quantités de tests est
80% de tests	unitaires, 15% de tests d'intégration, 5% de tests UI.
80% de tests	UI, 15% de tests d'intégration, 5% de tests unitaires.
80% de tests	d'intégration, 15% de tests unitaires, 5% de tests UI.

2023-2024 - R4-02 Qualité de développement - CC1 Veuillez coder votre numéro d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous. Nom et prénom : Durée totale : 42 mn (56 mn pour les tiers temps). Partie QCM sur 7 points. Aucun document ni dispositif électronique n'est autorisé. Le symbole "trèfle" sur une question indique qu'il peut être attendu plus d'une réponse correcte. Les réponses doivent être fournies uniquement sur les feuilles d'énoncé. Aucune autre réponse ne sera prise en compte dans la notation. Les cases doivent être entièrement coloriées en noir pour être prises en compte dans la notation. Pour modifier une réponse, vous pouvez utiliser un correcteur blanc, sans re-dessiner la case à cocher. Question 1 🌲 Bénéfices et limites des tests automatisés (TA) - Cochez les assertions vraies : Les TA permettent d'effectuer des changements dans le code en confiance. Les TA permettent de limiter le nombre de bugs. Les TA permettent de s'assurer qu'un programme fonctionne comme attendu. Les TA peuvent contenir des bugs. Question 2 🌲 Tests unitaires vs tests d'intégration - Cochez les assertions vraies : Les tests unitaires créent toujours les vrais collaborateurs pour tester un objet. Les tests unitaires consomment en général plus de ressources que les tests d'intégration. Les tests d'intégration s'exécutent en général plus rapidement que les tests unitaires. Les tests d'intégration vérifient le fonctionnement de "vrais" objets en interactions. Question 3 🌲 Suite de tests - Cochez les assertions vraies :

Une bonne suite de tests teste explicitement uniquement l'interface publique de l'objet testé.

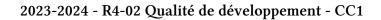
Une bonne suite de tests est une suite de tests où les tests dépendent les uns des autres.

Un test en échec indique en général un bug dans le programme principal.

Les tests des cas d'erreur suffisent pour tester la robustesse d'un programme.

Un test en erreur indique en général un bug dans le test.

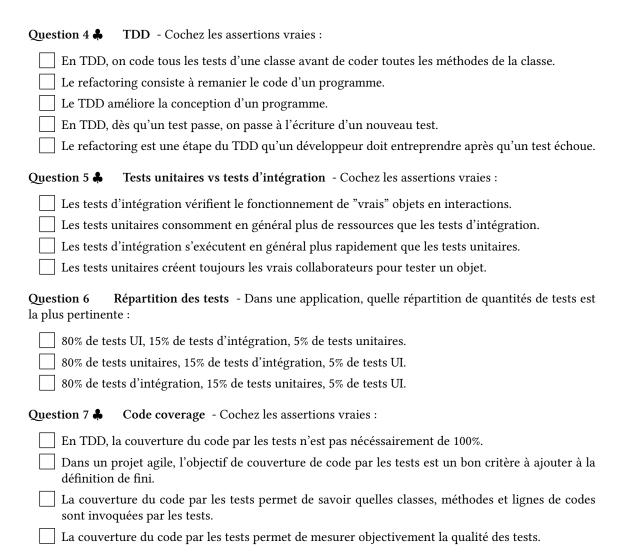
Question 4 ♣ TDD - Cochez les assertions vraies :
 En TDD, dès qu'un test passe, on passe à l'écriture d'un nouveau test. En TDD, on code tous les tests d'une classe avant de coder toutes les méthodes de la classe. Le TDD améliore la conception d'un programme. Le refactoring est une étape du TDD qu'un développeur doit entreprendre après qu'un test échoue Le refactoring consiste à remanier le code d'un programme.
Question 5 - Simuler des collaborateurs - Cochez les assertions vraies :
 Il existe des librairies pour créer des collaborateurs simulés dans à peu près tous les langages de programmation courants. Un collaborateur simulé est communément appelé en anglais "mock object". Simuler des collaborateurs dans un test permet en général de limiter le coût d'exécution d'un test. Il n'est pas possible de créer des collaborateurs simulés dans des tests en PHP. On utilise des collaborateurs simulés en général dans les tests d'intégration. Il n'est pas toujours possible de créer un collaborateur simulé "à la main" pour une classe donnée.
Question 6 Répartition des tests - Dans une application, quelle répartition de quantités de tests es la plus pertinente :
80% de tests d'intégration, 15% de tests unitaires, 5% de tests UI. 80% de tests UI, 15% de tests d'intégration, 5% de tests unitaires. 80% de tests unitaires, 15% de tests d'intégration, 5% de tests UI.
Question 7 & Code coverage - Cochez les assertions vraies :
 En TDD, la couverture du code par les tests n'est pas nécéssairement de 100%. Dans un projet agile, l'objectif de couverture de code par les tests est un bon critère à ajouter à la définition de fini. La couverture du code par les tests permet de savoir quelles classes, méthodes et lignes de codes sont invoquées par les tests.
La couverture du code par les tests permet de mesurer objectivement la qualité des tests.



\square_2 \square_2 \square_2 \square_2 \square_2 \square_2 \square_2		
\square_3 \square_3 \square_3 \square_3 \square_3 \square_3 \square_3	← Veuillez coder votre numéro	
	d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous.	
	dans la case el dessous.	
	Nom et prénom :	
Durée totale : 42 mn (56 mn pour les tiers temps). P Aucun document ni dispositif électronique n'est aut Le symbole "trèfle" sur une question indique qu'il po Les réponses doivent être fournies uniquement sur le prise en compte dans la notation. Les cases doivent être notation de la notation. Pour modifier une répons re-dessiner la case à cocher.	orisé. eut être attendu plus d'une réponse correcte. es feuilles d'énoncé. Aucune autre réponse ne sera re entièrement coloriées en noir pour être prises	
Question 1 Répartition des tests - Dans une applie la plus pertinente : 80% de tests d'intégration, 15% de tests unitaires, 80% de tests unitaires, 15% de tests d'intégration,		
80% de tests UI, 15% de tests d'intégration, 5% de tests unitaires.		
Question 2 . TDD - Cochez les assertions vraies :		
En TDD, on code tous les tests d'une classe avant	de coder toutes les méthodes de la classe.	
Le refactoring est une étape du TDD qu'un dévelo	oppeur doit entreprendre après qu'un test échoue.	
En TDD, dès qu'un test passe, on passe à l'écritur		
Le TDD améliore la conception d'un programme.		
Le refactoring consiste à remanier le code d'un pr	ogramme.	
	-8	
Question 3 - Suite de tests - Cochez les assertions v	vraies :	
Une bonne suite de tests teste explicitement uniqu	uement l'interface publique de l'objet testé.	
Les tests des cas d'erreur suffisent pour tester la r	obustesse d'un programme.	
Un test en échec indique en général un bug dans l	e programme principal.	
Un test en erreur indique en général un bug dans le test.		
Une bonne suite de tests est une suite de tests où les tests dépendent les uns des autres.		

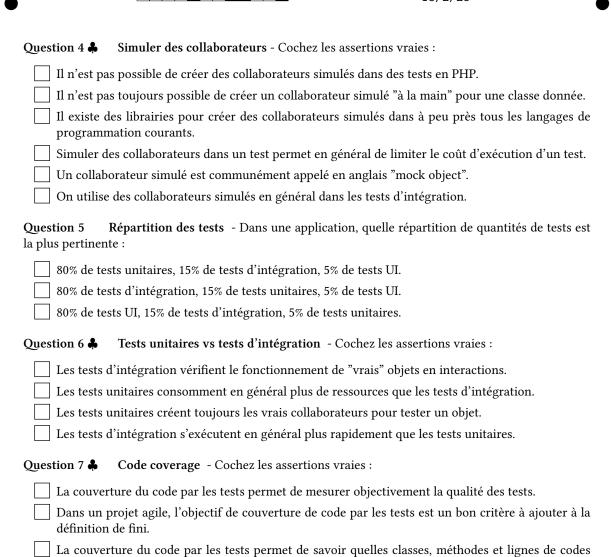
Question 4 - Simuler des collaborateurs - Cochez les assertions vraies :	
On utilise des collaborateurs simulés en général dans les tests d'intégration.	
Simuler des collaborateurs dans un test permet en général de limiter le coût d'exécution	d'un test.
Un collaborateur simulé est communément appelé en anglais "mock object".	
Il existe des librairies pour créer des collaborateurs simulés dans à peu près tous les la programmation courants.	ngages de
Il n'est pas toujours possible de créer un collaborateur simulé "à la main" pour une classe	e donnée.
Il n'est pas possible de créer des collaborateurs simulés dans des tests en PHP.	
Question 5 ♣ Tests unitaires vs tests d'intégration - Cochez les assertions vraies :	
Les tests unitaires consomment en général plus de ressources que les tests d'intégration.	
Les tests d'intégration s'exécutent en général plus rapidement que les tests unitaires.	
Les tests unitaires créent toujours les vrais collaborateurs pour tester un objet.	
Les tests d'intégration vérifient le fonctionnement de "vrais" objets en interactions.	
Question 6 & Code coverage - Cochez les assertions vraies :	
La couverture du code par les tests permet de savoir quelles classes, méthodes et lignes sont invoquées par les tests.	s de codes
La couverture du code par les tests permet de mesurer objectivement la qualité des tests.	
En TDD, la couverture du code par les tests n'est pas nécéssairement de 100%.	
Dans un projet agile, l'objectif de couverture de code par les tests est un bon critère à a définition de fini.	jouter à la
Question 7 & Bénéfices et limites des tests automatisés (TA) - Cochez les assertions vraie	s:
Les TA permettent de s'assurer qu'un programme fonctionne comme attendu.	
Les TA permettent d'effectuer des changements dans le code en confiance.	
Les TA permettent de limiter le nombre de bugs.	
Les TA peuvent contenir des bugs.	

$\begin{array}{cccccccccccccccccccccccccccccccccccc$		
	← Veuillez coder votre numéro	
	d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous.	
	uans la case el dessous.	
	Nom et prénom :	
Durée totale : 42 mn (56 mn pour les tiers temps). Pa Aucun document ni dispositif électronique n'est auto Le symbole "trèfle" sur une question indique qu'il pe Les réponses doivent être fournies uniquement sur le prise en compte dans la notation. Les cases doivent être en compte dans la notation. Pour modifier une réponse re-dessiner la case à cocher.	orisé. Put être attendu plus d'une réponse correcte. Is feuilles d'énoncé. Aucune autre réponse ne sera re entièrement coloriées en noir pour être prises	
Question 1 ♣ Bénéfices et limites des tests automatisés (TA) - Cochez les assertions vraies : □ Les TA peuvent contenir des bugs. □ Les TA permettent de s'assurer qu'un programme fonctionne comme attendu. □ Les TA permettent de limiter le nombre de bugs.		
Les TA permettent d'effectuer des changements de	ans le code en confiance.	
Question 2 & Simuler des collaborateurs - Cochez le	es assertions vraies :	
On utilise des collaborateurs simulés en général d	ans les tests d'intégration.	
☐ Il n'est pas possible de créer des collaborateurs sir	nulés dans des tests en PHP.	
Il existe des librairies pour créer des collaborateurs simulés dans à peu près tous les langages de programmation courants.		
Simuler des collaborateurs dans un test permet en général de limiter le coût d'exécution d'un test.		
Un collaborateur simulé est communément appelé en anglais "mock object".		
☐ Il n'est pas toujours possible de créer un collabora	teur simulé "à la main" pour une classe donnée.	
Question 3 - Suite de tests - Cochez les assertions v	raies:	
Un test en erreur indique en général un bug dans	le test.	
Un test en échec indique en général un bug dans le programme principal.		
Une bonne suite de tests est une suite de tests où les tests dépendent les uns des autres.		
Les tests des cas d'erreur suffisent pour tester la robustesse d'un programme.		
Une bonne suite de tests teste explicitement uniquement l'interface publique de l'objet testé.		





\square_2 \square_2 \square_2 \square_2 \square_2 \square_2 \square_2	
\square_3 \square_3 \square_3 \square_3 \square_3 \square_3 \square_3	← Veuillez coder votre numéro
	d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous.
	dans la case el dessous.
	Nom et prénom :
Durée totale : 42 mn (56 mn pour les tiers temps). Aucun document ni dispositif électronique n'est. Le symbole "trèfle" sur une question indique qu'i Les réponses doivent être fournies uniquement su prise en compte dans la notation. Les cases doivent en compte dans la notation. Pour modifier une réporte-dessiner la case à cocher.	autorisé. il peut être attendu plus d'une réponse correcte. ur les feuilles d'énoncé. Aucune autre réponse ne sera t être entièrement coloriées en noir pour être prises
Question 1 ♣ TDD - Cochez les assertions vraies	s :
Le refactoring est une étape du TDD qu'un dév	veloppeur doit entreprendre après qu'un test échoue.
Le refactoring consiste à remanier le code d'ur	n programme.
En TDD, dès qu'un test passe, on passe à l'écri	iture d'un nouveau test.
En TDD, on code tous les tests d'une classe av	ant de coder toutes les méthodes de la classe.
Le TDD améliore la conception d'un programme	me.
Question 2 - Suite de tests - Cochez les assertion	ns vraies :
Un test en échec indique en général un bug da	ns le programme principal.
Une bonne suite de tests teste explicitement u	
Un test en erreur indique en général un bug da	
Les tests des cas d'erreur suffisent pour tester	
Une bonne suite de tests est une suite de tests	. 0
_	•
Question 3 4 Bénéfices et limites des tests autor	matisés (TA) - Cochez les assertions vraies :
Les TA peuvent contenir des bugs.	
Les TA permettent de s'assurer qu'un program	nme fonctionne comme attendu.
Les TA permettent de limiter le nombre de bug	gs.
Les TA permettent d'effectuer des changement	ts dans le code en confiance.



En TDD, la couverture du code par les tests n'est pas nécéssairement de 100%.

sont invoquées par les tests.

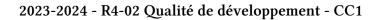
3 3 3 3 3 3 3 3	← Veuillez coder votre numéro
	d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous.
	Nom et prénom :
Durée totale : 42 mn (56 mn pour les tiers temps) Aucun document ni dispositif électronique n'est a Le symbole "trèfle" sur une question indique qu'i Les réponses doivent être fournies uniquement su prise en compte dans la notation. Les cases doivent en compte dans la notation. Pour modifier une répor re-dessiner la case à cocher.	autorisé. l peut être attendu plus d'une réponse correcte. ir les feuilles d'énoncé. Aucune autre réponse ne sera : être entièrement coloriées en noir pour être prises
Question 1 Répartition des tests - Dans une appla plus pertinente : 80% de tests unitaires, 15% de tests d'intégration 80% de tests d'intégration, 15% de tests unitaires 80% de tests UI, 15% de tests d'intégration, 5% de	es, 5% de tests UI.
Question 2 - Simuler des collaborateurs - Coche	ez les assertions vraies :
Simuler des collaborateurs dans un test permet	t en général de limiter le coût d'exécution d'un test.
	ateurs simulés dans à peu près tous les langages de
Il n'est pas possible de créer des collaborateurs	simulés dans des tests en PHP.
On utilise des collaborateurs simulés en généra	al dans les tests d'intégration.
Il n'est pas toujours possible de créer un collab	orateur simulé "à la main" pour une classe donnée.
Un collaborateur simulé est communément app	pelé en anglais "mock object".
Question 3 🌲 Tests unitaires vs tests d'intégration	on - Cochez les assertions vraies :
Les tests d'intégration s'exécutent en général p	olus rapidement que les tests unitaires.
Les tests unitaires consomment en général plus	s de ressources que les tests d'intégration.
Les tests d'intégration vérifient le fonctionnem	nent de "vrais" objets en interactions.
Les tests unitaires créent toujours les vrais coll	aborateurs pour tester un objet.

Question 4 🐥	Bénéfices et limites des tests automatisés (TA) - Cochez les assertions vraies :
Les TA pe	uvent contenir des bugs.
Les TA pe	rmettent de s'assurer qu'un programme fonctionne comme attendu.
Les TA pe	rmettent d'effectuer des changements dans le code en confiance.
Les TA pe	rmettent de limiter le nombre de bugs.
Question 5 🌲	Code coverage - Cochez les assertions vraies :
Dans un p	rojet agile, l'objectif de couverture de code par les tests est un bon critère à ajouter à la de fini.
En TDD, l	a couverture du code par les tests n'est pas nécéssairement de 100%.
	ture du code par les tests permet de savoir quelles classes, méthodes et lignes de codes quées par les tests.
La couver	rure du code par les tests permet de mesurer objectivement la qualité des tests.
Question 6 🌲	TDD - Cochez les assertions vraies :
Le refacto	ring est une étape du TDD qu'un développeur doit entreprendre après qu'un test échoue
Le refacto	ring consiste à remanier le code d'un programme.
Le TDD at	néliore la conception d'un programme.
En TDD, d	ès qu'un test passe, on passe à l'écriture d'un nouveau test.
En TDD, o	n code tous les tests d'une classe avant de coder toutes les méthodes de la classe.
Question 7 🌲	Suite de tests - Cochez les assertions vraies :
Une bonn	e suite de tests teste explicitement uniquement l'interface publique de l'objet testé.
Les tests d	es cas d'erreur suffisent pour tester la robustesse d'un programme.
Un test en	échec indique en général un bug dans le programme principal.
Une bonn	e suite de tests est une suite de tests où les tests dépendent les uns des autres.
Un test en	erreur indique en général un bug dans le test.



0 0				
Durée totale : 42 mn (56 mn pour les tiers temps). Partie QCM sur 7 points. Aucun document ni dispositif électronique n'est autorisé. Le symbole "trèfle" sur une question indique qu'il peut être attendu plus d'une réponse correcte. Les réponses doivent être fournies uniquement sur les feuilles d'énoncé. Aucune autre réponse ne sera prise en compte dans la notation. Les cases doivent être entièrement coloriées en noir pour être prises en compte dans la notation. Pour modifier une réponse, vous pouvez utiliser un correcteur blanc, sans re-dessiner la case à cocher.				
Question 1 & Code coverage - Cochez les assertions	vraies:			
Dans un projet agile, l'objectif de couverture de c	ode par les tests est un bon critère à ajouter à la			
définition de fini.	200 m d d d d d d d d d d d d d d d d d d			
En TDD, la couverture du code par les tests n'est pas nécéssairement de 100%.				
La couverture du code par les tests permet de mes La couverture du code par les tests permet de sav sont invoquées par les tests.				
Question 2 - Suite de tests - Cochez les assertions v	raies :			
☐ Une bonne suite de tests teste explicitement unique☐ Les tests des cas d'erreur suffisent pour tester la re				
Une bonne suite de tests est une suite de tests où l	es tests dépendent les uns des autres.			
Un test en erreur indique en général un bug dans				
Un test en échec indique en général un bug dans l	e programme principal.			
Question 3 . Simuler des collaborateurs - Cochez le	es assertions vraies :			
☐ Il n'est pas possible de créer des collaborateurs sin	nulés dans des tests en PHP.			
Il existe des librairies pour créer des collaborateu programmation courants.	rs simulés dans à peu près tous les langages de			
Simuler des collaborateurs dans un test permet en	général de limiter le coût d'exécution d'un test.			
Un collaborateur simulé est communément appelé	en anglais "mock object".			
☐ Il n'est pas toujours possible de créer un collabora	teur simulé "à la main" pour une classe donnée.			
On utilise des collaborateurs simulés en général dans les tests d'intégration.				

Question 4 ૈ	Tests unitaires vs tests d'intégration - Cochez les assertions vraies :
Les tests	d'intégration vérifient le fonctionnement de "vrais" objets en interactions.
Les tests	unitaires créent toujours les vrais collaborateurs pour tester un objet.
Les tests	unitaires consomment en général plus de ressources que les tests d'intégration.
Les tests	d'intégration s'exécutent en général plus rapidement que les tests unitaires.
Question 5 la plus pertiner	Répartition des tests - Dans une application, quelle répartition de quantités de tests est ate :
80% de te	ests unitaires, 15% de tests d'intégration, 5% de tests UI.
80% de te	sts d'intégration, 15% de tests unitaires, 5% de tests UI.
80% de te	sts UI, 15% de tests d'intégration, 5% de tests unitaires.
Question 6 ૈ	TDD - Cochez les assertions vraies :
Le TDD a	améliore la conception d'un programme.
En TDD,	dès qu'un test passe, on passe à l'écriture d'un nouveau test.
Le refacte	oring est une étape du TDD qu'un développeur doit entreprendre après qu'un test échoue.
Le refact	oring consiste à remanier le code d'un programme.
En TDD,	on code tous les tests d'une classe avant de coder toutes les méthodes de la classe.
Question 7 🌲	Bénéfices et limites des tests automatisés (TA) - Cochez les assertions vraies :
Les TA p	euvent contenir des bugs.
Les TA p	ermettent de limiter le nombre de bugs.
Les TA p	ermettent de s'assurer qu'un programme fonctionne comme attendu.
Les TA p	ermettent d'effectuer des changements dans le code en confiance.

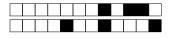


33333333	← Veuillez coder votre numéro d'étudiant ci-contre, et écrire votre nom			
$\square 4$	dans la case ci-dessous.			
555555				
	Nom et prénom :			
999999				
Durée totale : 42 mn (56 mn pour les tiers temps). Partie QCM sur 7 points. Aucun document ni dispositif électronique n'est autorisé. Le symbole "trèfle" sur une question indique qu'il peut être attendu plus d'une réponse correcte. Les réponses doivent être fournies uniquement sur les feuilles d'énoncé. Aucune autre réponse ne sera prise en compte dans la notation. Les cases doivent être entièrement coloriées en noir pour être prises en compte dans la notation. Pour modifier une réponse, vous pouvez utiliser un correcteur blanc, sans re-dessiner la case à cocher.				
Question 1 Répartition des tests - Dans une applic la plus pertinente : 80% de tests unitaires, 15% de tests d'intégration, 5	ration, quelle répartition de quantités de tests est			
80% de tests d'intégration, 15% de tests unitaires, 5% de tests UI. 80% de tests UI, 15% de tests d'intégration, 5% de tests unitaires.				
ovi de tests et, 13% de tests à linegration, 3% de t	ests diffidires.			
Question 2 🌲 Bénéfices et limites des tests automat	isés (TA) - Cochez les assertions vraies :			
Les TA permettent de limiter le nombre de bugs.				
Les TA permettent d'effectuer des changements de	ans le code en confiance.			
Les TA peuvent contenir des bugs.				
Les TA permettent de s'assurer qu'un programme	fonctionne comme attendu.			
Question 3 & Simuler des collaborateurs - Cochez le	es assertions vraies :			
Simuler des collaborateurs dans un test permet en	général de limiter le coût d'exécution d'un test.			
Un collaborateur simulé est communément appelé	en anglais "mock object".			
On utilise des collaborateurs simulés en général d	ans les tests d'intégration.			
Il existe des librairies pour créer des collaborateu programmation courants.	ars simulés dans à peu près tous les langages de			
☐ Il n'est pas possible de créer des collaborateurs sir	nulés dans des tests en PHP.			
Il n'est pas toujours possible de créer un collabora	teur simulé "à la main" pour une classe donnée.			

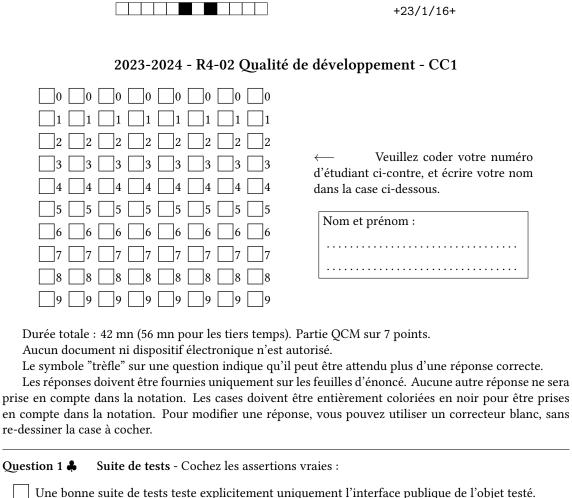
Questi	n 4 🌲 Tests unitaires vs tests d'intégration - Cochez les assertions vraies :
	es tests d'intégration vérifient le fonctionnement de "vrais" objets en interactions.
	es tests unitaires créent toujours les vrais collaborateurs pour tester un objet.
	es tests unitaires consomment en général plus de ressources que les tests d'intégration.
	es tests d'intégration s'exécutent en général plus rapidement que les tests unitaires.
Questic	on 5 🌲 Suite de tests - Cochez les assertions vraies :
	es tests des cas d'erreur suffisent pour tester la robustesse d'un programme.
	n test en échec indique en général un bug dans le programme principal.
	n test en erreur indique en général un bug dans le test.
	ne bonne suite de tests teste explicitement uniquement l'interface publique de l'objet testé.
	ne bonne suite de tests est une suite de tests où les tests dépendent les uns des autres.
Questic	on 6 ♣ TDD - Cochez les assertions vraies :
	e refactoring est une étape du TDD qu'un développeur doit entreprendre après qu'un test échoue
	e refactoring consiste à remanier le code d'un programme.
	e TDD améliore la conception d'un programme.
E	n TDD, on code tous les tests d'une classe avant de coder toutes les méthodes de la classe.
E	n TDD, dès qu'un test passe, on passe à l'écriture d'un nouveau test.
Questic	on 7 ♣ Code coverage - Cochez les assertions vraies :
	ans un projet agile, l'objectif de couverture de code par les tests est un bon critère à ajouter à la éfinition de fini.
E	n TDD, la couverture du code par les tests n'est pas nécéssairement de 100%.
	a couverture du code par les tests permet de savoir quelles classes, méthodes et lignes de codes ont invoquées par les tests.
	a couverture du code par les tests permet de mesurer objectivement la qualité des tests.



	000	00	<u></u> 0 ∟0	00	
	1 1 1	<u> </u>	<u> </u>	1 1	
	2 2 2	\square_2 \square_2		2 2	
	3 3 3	3 3	3 3	3 3	← Veuillez coder votre numéro
	$\frac{1}{4} \frac{1}{4} \frac{1}{4}$			4 4	d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous.
	5			5	
	6			5 □6	Nom et prénom :
	7			7	
				/ <u> </u> /	
	8			3 <u> </u> 8	
	999	<u></u> 9 <u></u> 9	<u></u> 9 <u></u>	99	
Aucun Le syn Les rép prise en c en compte	i document n nbole "trèfle" ponses doiver ompte dans l	i dispositif sur une qu nt être fourn la notation. ation. Pour	électroniq estion ind nies uniqu Les case	que n'est aut lique qu'il po nement sur le s doivent êtr	artie QCM sur 7 points. orisé. cut être attendu plus d'une réponse correcte. s feuilles d'énoncé. Aucune autre réponse ne sera ce entièrement coloriées en noir pour être prises e, vous pouvez utiliser un correcteur blanc, sans
Les	e bonne suite tests des cas test en erreur	de tests tes d'erreur su indique en	te explicit ffisent pou général u	ur tester la r un bug dans	nement l'interface publique de l'objet testé. Obustesse d'un programme.
Un	test en échec	indique en	général u	ın bug dans l	e programme principal.
Question la plus per	-	ition des te	ests - Dar	ıs une applic	ation, quelle répartition de quantités de tests est
80%	de tests unit	aires, 15% d	le tests d'i	intégration,	5% de tests UI.
80%	de tests UI, 1	15% de tests	d'intégra	ation, 5% de t	ests unitaires.
80%	de tests d'in	tégration, 1	5% de test	ts unitaires,	5% de tests UI.
Question	3 🌲 Simu	ler des coll	aborateui	rs - Cochez l	es assertions vraies :
Sim	uler des colla	borateurs d	lans un te	est permet er	général de limiter le coût d'exécution d'un test.
On	utilise des co	llaborateur	s simulés	en général d	ans les tests d'intégration.
Un	collaborateur	simulé est	commune	ément appel	en anglais "mock object".
	xiste des libra grammation o	_	créer des	collaborate	rrs simulés dans à peu près tous les langages de
☐ Il n'	est pas toujo	urs possible	e de créer	un collabora	teur simulé "à la main" pour une classe donnée.
Il n'est pas possible de créer des collaborateurs simulés dans des tests en PHP.					

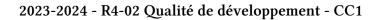


Question 4 A TDD - Cochez les assertions vraies :
Le TDD améliore la conception d'un programme.
Le refactoring consiste à remanier le code d'un programme.
En TDD, on code tous les tests d'une classe avant de coder toutes les méthodes de la classe.
En TDD, dès qu'un test passe, on passe à l'écriture d'un nouveau test.
Le refactoring est une étape du TDD qu'un développeur doit entreprendre après qu'un test échoue
Question 5 ♣ Code coverage - Cochez les assertions vraies :
La couverture du code par les tests permet de savoir quelles classes, méthodes et lignes de codes sont invoquées par les tests.
Dans un projet agile, l'objectif de couverture de code par les tests est un bon critère à ajouter à la définition de fini.
La couverture du code par les tests permet de mesurer objectivement la qualité des tests.
En TDD, la couverture du code par les tests n'est pas nécéssairement de 100%.
Question 6 - Cochez les assertions vraies :
Les tests unitaires créent toujours les vrais collaborateurs pour tester un objet.
Les tests d'intégration s'exécutent en général plus rapidement que les tests unitaires.
Les tests d'intégration vérifient le fonctionnement de "vrais" objets en interactions.
Les tests unitaires consomment en général plus de ressources que les tests d'intégration.
Question 7 - Bénéfices et limites des tests automatisés (TA) - Cochez les assertions vraies :
Les TA peuvent contenir des bugs.
Les TA permettent d'effectuer des changements dans le code en confiance.
Les TA permettent de s'assurer qu'un programme fonctionne comme attendu.
Les TA permettent de limiter le nombre de bugs.

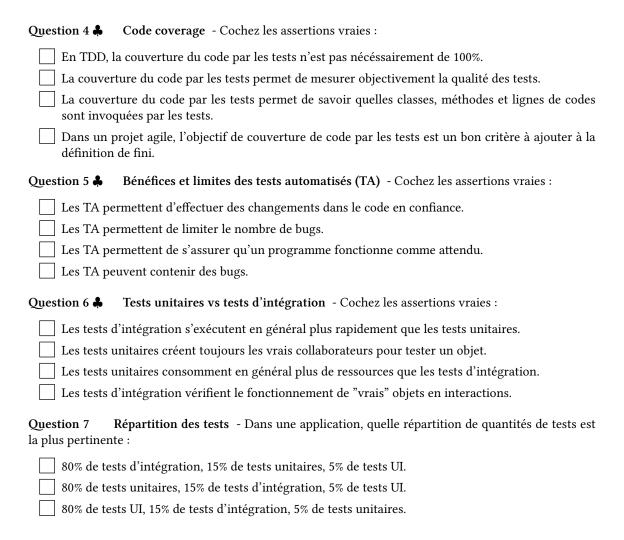


Question 1 Suite de tests - Cochez les assertions vraies :
Une bonne suite de tests teste explicitement uniquement l'interface publique de l'objet testé.
Une bonne suite de tests est une suite de tests où les tests dépendent les uns des autres.
Les tests des cas d'erreur suffisent pour tester la robustesse d'un programme.
Un test en erreur indique en général un bug dans le test.
Un test en échec indique en général un bug dans le programme principal.
Question 2 A TDD - Cochez les assertions vraies :
Le refactoring consiste à remanier le code d'un programme.
Le refactoring est une étape du TDD qu'un développeur doit entreprendre après qu'un test échoue.
En TDD, on code tous les tests d'une classe avant de coder toutes les méthodes de la classe.
En TDD, dès qu'un test passe, on passe à l'écriture d'un nouveau test.
Le TDD améliore la conception d'un programme.
Question 3 🌲 Bénéfices et limites des tests automatisés (TA) - Cochez les assertions vraies :
Les TA permettent d'effectuer des changements dans le code en confiance.
Les TA peuvent contenir des bugs.
Les TA permettent de limiter le nombre de bugs.
Les TA permettent de s'assurer qu'un programme fonctionne comme attendu.

Question 4 ♣ Tests unitaires vs tests d'intégration - Cochez les assertions vraies :
Les tests d'intégration vérifient le fonctionnement de "vrais" objets en interactions. Les tests unitaires consomment en général plus de ressources que les tests d'intégration. Les tests d'intégration s'exécutent en général plus rapidement que les tests unitaires. Les tests unitaires créent toujours les vrais collaborateurs pour tester un objet.
Question 5 & Code coverage - Cochez les assertions vraies :
 La couverture du code par les tests permet de mesurer objectivement la qualité des tests. En TDD, la couverture du code par les tests n'est pas nécéssairement de 100%. La couverture du code par les tests permet de savoir quelles classes, méthodes et lignes de code sont invoquées par les tests. Dans un projet agile, l'objectif de couverture de code par les tests est un bon critère à ajouter à l'appendit de couverture de code par les tests est un bon critère à ajouter à l'appendit de couverture de code par les tests est un bon critère à ajouter à l'appendit de couverture de code par les tests est un bon critère à ajouter à l'appendit de code par les tests est un bon critère à ajouter à l'appendit de code par les tests est un bon critère à ajouter à l'appendit de code par les tests est un bon critère à ajouter à l'appendit de code par les tests est un bon critère à ajouter à l'appendit de code par les tests est un bon critère à ajouter à l'appendit de code par les tests est un bon critère à ajouter à l'appendit de code par les tests est un bon critère à ajouter à l'appendit de code par les tests est un bon critère à ajouter à l'appendit de code par les tests est un bon critère à ajouter à l'appendit de code par les tests est un bon critère à ajouter à l'appendit de code par les tests est un bon critère à ajouter à l'appendit de code par les tests est un bon critère à ajouter à l'appendit de code par les tests est un bon critère à ajouter à l'appendit de code par les tests est un bon critère à l'appendit de code par les tests est un bon critère à l'appendit de code par les tests est un bon critère à l'appendit de code par les tests est un bon critère de code par les tests est un bon critère de code par les tests est un bon critère à l'appendit de code par les tests est un bon critère de code par les tests est un bon critère à ajouter à l'appendit de code par les tests est un bon critère à ajouter l'appendit de code par les tests est un bon critère de code par les
définition de fini. Question 6 Simuler des collaborateurs - Cochez les assertions vraies :
 Il n'est pas toujours possible de créer un collaborateur simulé "à la main" pour une classe donnée Il existe des librairies pour créer des collaborateurs simulés dans à peu près tous les langages de programmation courants.
 Il n'est pas possible de créer des collaborateurs simulés dans des tests en PHP. ☐ On utilise des collaborateurs simulés en général dans les tests d'intégration. ☐ Simuler des collaborateurs dans un test permet en général de limiter le coût d'exécution d'un test ☐ Un collaborateur simulé est communément appelé en anglais "mock object".
Question 7 Répartition des tests - Dans une application, quelle répartition de quantités de tests et la plus pertinente :
80% de tests UI, 15% de tests d'intégration, 5% de tests unitaires. 80% de tests unitaires, 15% de tests d'intégration, 5% de tests UI. 80% de tests d'intégration, 15% de tests unitaires, 5% de tests UI.

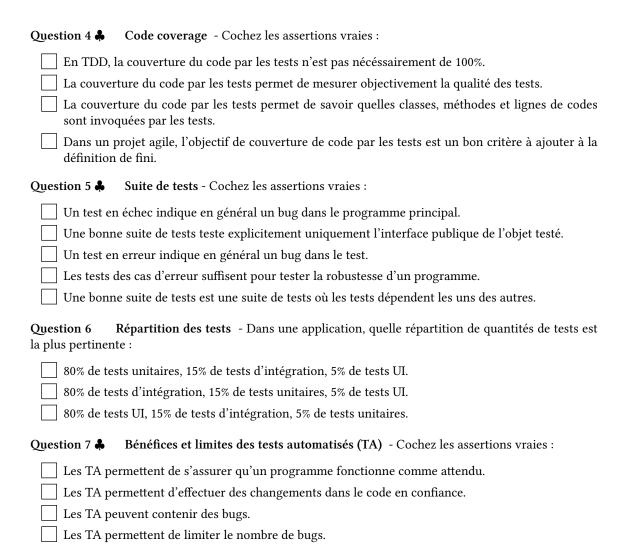


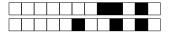
\square_2 \square_2 \square_2 \square_2 \square_2 \square_2 \square_2				
\square_3 \square_3 \square_3 \square_3 \square_3 \square_3 \square_3	← Veuillez coder votre numéro			
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous.			
	dans in case of desired.			
	Nom et prénom :			
Durée totale : 42 mn (56 mn pour les tiers temps). Partie QCM sur 7 points. Aucun document ni dispositif électronique n'est autorisé. Le symbole "trèfle" sur une question indique qu'il peut être attendu plus d'une réponse correcte. Les réponses doivent être fournies uniquement sur les feuilles d'énoncé. Aucune autre réponse ne sera prise en compte dans la notation. Les cases doivent être entièrement coloriées en noir pour être prises en compte dans la notation. Pour modifier une réponse, vous pouvez utiliser un correcteur blanc, sans re-dessiner la case à cocher.				
Question 1 - Suite de tests - Cochez les assertions v	raies :			
Un test en échec indique en général un bug dans le	e programme principal.			
Une bonne suite de tests teste explicitement uniqu	ement l'interface publique de l'objet testé.			
Les tests des cas d'erreur suffisent pour tester la robustesse d'un programme.				
Une bonne suite de tests est une suite de tests où les tests dépendent les uns des autres.				
Un test en erreur indique en général un bug dans l	e test.			
Question 2 - Cochez le	es assertions vraies :			
Simuler des collaborateurs dans un test permet en	général de limiter le coût d'exécution d'un test.			
☐ Il n'est pas toujours possible de créer un collabora	teur simulé "à la main" pour une classe donnée.			
On utilise des collaborateurs simulés en général dans les tests d'intégration.				
Il existe des librairies pour créer des collaborateu programmation courants.	rs simulés dans à peu près tous les langages de			
☐ Il n'est pas possible de créer des collaborateurs sin	nulés dans des tests en PHP.			
Un collaborateur simulé est communément appelé	en anglais "mock object".			
Question 3 4 TDD - Cochez les assertions vraies :				
Le TDD améliore la conception d'un programme.				
En TDD, on code tous les tests d'une classe avant	de coder toutes les méthodes de la classe.			
En TDD, dès qu'un test passe, on passe à l'écriture				
Le refactoring est une étape du TDD qu'un dévelo				
Le refactoring consiste à remanier le code d'un pro				





0 0			
Durée totale : 42 mn (56 mn pour les tiers temps). Partie QCM sur 7 points. Aucun document ni dispositif électronique n'est autorisé. Le symbole "trèfle" sur une question indique qu'il peut être attendu plus d'une réponse correcte. Les réponses doivent être fournies uniquement sur les feuilles d'énoncé. Aucune autre réponse ne sera prise en compte dans la notation. Les cases doivent être entièrement coloriées en noir pour être prises en compte dans la notation. Pour modifier une réponse, vous pouvez utiliser un correcteur blanc, sans re-dessiner la case à cocher.			
Question 1 ♣ TDD - Cochez les assertions vraies : Le refactoring est une étape du TDD qu'un dévelo Le refactoring consiste à remanier le code d'un pr En TDD, dès qu'un test passe, on passe à l'écritur Le TDD améliore la conception d'un programme. En TDD, on code tous les tests d'une classe avant	rogramme. re d'un nouveau test.		
Question 2 ♣ Tests unitaires vs tests d'intégration Les tests d'intégration vérifient le fonctionnemen Les tests d'intégration s'exécutent en général plus Les tests unitaires consomment en général plus d Les tests unitaires créent toujours les vrais collab	t de "vrais" objets en interactions. s rapidement que les tests unitaires. e ressources que les tests d'intégration.		
Question 3 Simuler des collaborateurs - Cochez l Simuler des collaborateurs dans un test permet er Il existe des librairies pour créer des collaborateur programmation courants.	n général de limiter le coût d'exécution d'un test.		
 On utilise des collaborateurs simulés en général d Un collaborateur simulé est communément appel Il n'est pas toujours possible de créer un collaborateur Il n'est pas possible de créer des collaborateurs si 	é en anglais "mock object". ateur simulé "à la main" pour une classe donnée.		





0 0	← Veuillez coder votre numéro d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous. Nom et prénom :
Durée totale : 42 mn (56 mn pour les tiers temps). P Aucun document ni dispositif électronique n'est aut Le symbole "trèfle" sur une question indique qu'il pe Les réponses doivent être fournies uniquement sur le prise en compte dans la notation. Les cases doivent être en compte dans la notation. Pour modifier une répons re-dessiner la case à cocher.	corisé. eut être attendu plus d'une réponse correcte. es feuilles d'énoncé. Aucune autre réponse ne sera re entièrement coloriées en noir pour être prises
Question 1 ♣ Suite de tests - Cochez les assertions v ☐ Une bonne suite de tests est une suite de tests où ☐ Les tests des cas d'erreur suffisent pour tester la r ☐ Un test en erreur indique en général un bug dans ☐ Un test en échec indique en général un bug dans ☐ Une bonne suite de tests teste explicitement unique	les tests dépendent les uns des autres. obustesse d'un programme. le test. le programme principal.
Question 2 ♣ Bénéfices et limites des tests automat Les TA peuvent contenir des bugs. Les TA permettent d'effectuer des changements d Les TA permettent de s'assurer qu'un programme Les TA permettent de limiter le nombre de bugs.	
Question 3 ♣ Simuler des collaborateurs - Cochez l Il n'est pas toujours possible de créer un collabora Un collaborateur simulé est communément appel Simuler des collaborateurs dans un test permet er Il n'est pas possible de créer des collaborateurs simulés en général d Il existe des librairies pour créer des collaborateur programmation courants.	ateur simulé "à la main" pour une classe donnée. é en anglais "mock object". n général de limiter le coût d'exécution d'un test. mulés dans des tests en PHP. lans les tests d'intégration.

Question 4 Répartiti la plus pertinente :	on des tests - Dans une application, quelle répartition de quantités de tests est
80% de tests UI, 15	% de tests d'intégration, 5% de tests unitaires.
80% de tests unitai	res, 15% de tests d'intégration, 5% de tests UI.
80% de tests d'intég	gration, 15% de tests unitaires, 5% de tests UI.
Question 5 4 TDD -	Cochez les assertions vraies :
En TDD, dès qu'un	test passe, on passe à l'écriture d'un nouveau test.
Le refactoring cons	iste à remanier le code d'un programme.
En TDD, on code to	ous les tests d'une classe avant de coder toutes les méthodes de la classe.
Le TDD améliore la	a conception d'un programme.
Le refactoring est ι	ne étape du TDD qu'un développeur doit entreprendre après qu'un test échoue.
Question 6 4 Tests u	nitaires vs tests d'intégration - Cochez les assertions vraies :
Les tests d'intégrat	ion vérifient le fonctionnement de "vrais" objets en interactions.
Les tests unitaires	créent toujours les vrais collaborateurs pour tester un objet.
Les tests d'intégrat	ion s'exécutent en général plus rapidement que les tests unitaires.
Les tests unitaires	consomment en général plus de ressources que les tests d'intégration.
Question 7 4 Code co	overage - Cochez les assertions vraies :
En TDD, la couver	ture du code par les tests n'est pas nécéssairement de 100%.
Dans un projet agi définition de fini.	le, l'objectif de couverture de code par les tests est un bon critère à ajouter à la
La couverture du c	ode par les tests permet de mesurer objectivement la qualité des tests.
La couverture du c	ode par les tests permet de savoir quelles classes, méthodes et lignes de codes

......

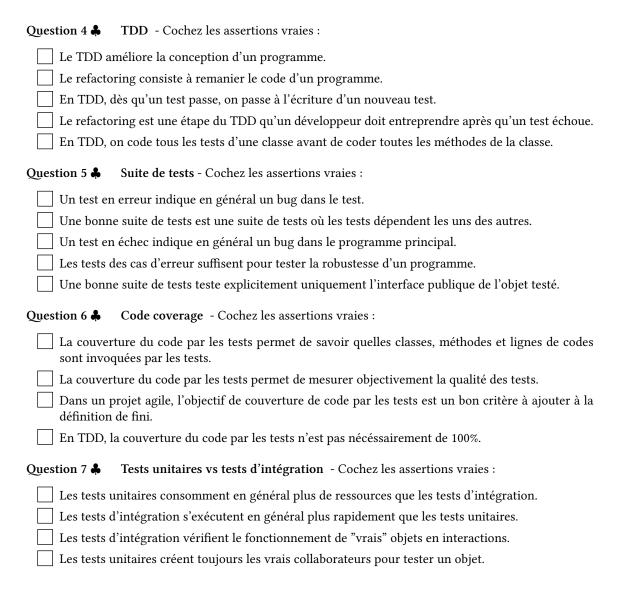
Durée totale : 42 mn (56 mn pour les tiers temps). Partie QCM sur 7 points.

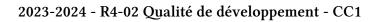
Aucun document ni dispositif électronique n'est autorisé.

Le symbole "trèfle" sur une question indique qu'il peut être attendu plus d'une réponse correcte.

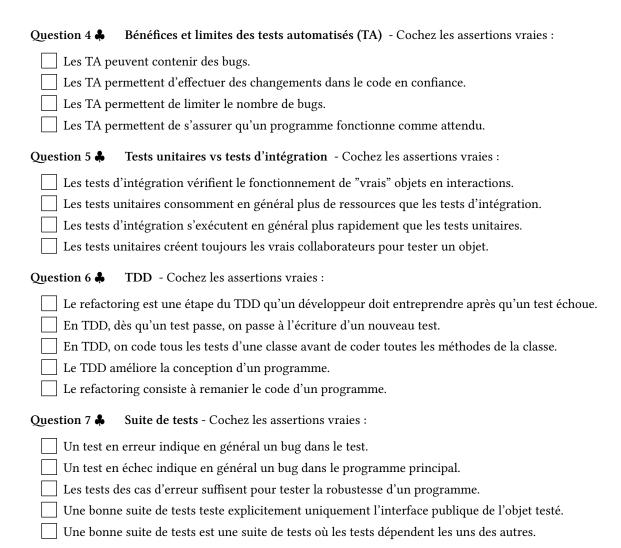
Les réponses doivent être fournies uniquement sur les feuilles d'énoncé. Aucune autre réponse ne sera prise en compte dans la notation. Les cases doivent être entièrement coloriées en noir pour être prises en compte dans la notation. Pour modifier une réponse, vous pouvez utiliser un correcteur blanc, sans re-dessiner la case à cocher.

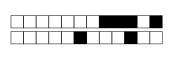
Question 1 & Bénéfices et limites des tests automatisés (TA) - Cochez les assertions vraies :
Les TA permettent de limiter le nombre de bugs.
Les TA peuvent contenir des bugs.
Les TA permettent d'effectuer des changements dans le code en confiance.
Les TA permettent de s'assurer qu'un programme fonctionne comme attendu.
Question 2 - Simuler des collaborateurs - Cochez les assertions vraies :
Simuler des collaborateurs dans un test permet en général de limiter le coût d'exécution d'un test.
Il existe des librairies pour créer des collaborateurs simulés dans à peu près tous les langages de programmation courants.
On utilise des collaborateurs simulés en général dans les tests d'intégration.
☐ Il n'est pas possible de créer des collaborateurs simulés dans des tests en PHP.
Il n'est pas toujours possible de créer un collaborateur simulé "à la main" pour une classe donnée.
Un collaborateur simulé est communément appelé en anglais "mock object".
Question 3 Répartition des tests - Dans une application, quelle répartition de quantités de tests est la plus pertinente :
80% de tests UI, 15% de tests d'intégration, 5% de tests unitaires.
80% de tests d'intégration, 15% de tests unitaires, 5% de tests UI.
80% de tests unitaires, 15% de tests d'intégration, 5% de tests UI.





\square_2 \square_2 \square_2 \square_2 \square_2 \square_2 \square_2	** '11
	 Veuillez coder votre numéro d'étudiant ci-contre, et écrire votre nom
$\boxed{}4 \ \boxed{}4 \ \boxed{}4 \ \boxed{}4 \ \boxed{}4 \ \boxed{}4 \ \boxed{}4 \ \boxed{}4$	dans la case ci-dessous.
555555	N , ,
	Nom et prénom :
8 8 8 8 8 8 8 8	
99999	
Durée totale : 42 mn (56 mn pour les tiers temps). F Aucun document ni dispositif électronique n'est aut Le symbole "trèfle" sur une question indique qu'il p Les réponses doivent être fournies uniquement sur le prise en compte dans la notation. Les cases doivent êt en compte dans la notation. Pour modifier une répons re-dessiner la case à cocher.	torisé. leut être attendu plus d'une réponse correcte. es feuilles d'énoncé. Aucune autre réponse ne sera tre entièrement coloriées en noir pour être prises
Question 1 Répartition des tests - Dans une appli la plus pertinente : 80% de tests d'intégration, 15% de tests unitaires,	cation, quelle répartition de quantités de tests est
80% de tests UI, 15% de tests d'intégration, 5% de	
80% de tests unitaires, 15% de tests d'intégration,	
50% de tests difitalies, 13% de tests d'integration,	5% de tests 01.
Question 2 - Simuler des collaborateurs - Cochez	les assertions vraies :
☐ Il n'est pas toujours possible de créer un collabor	ateur simulé "à la main" pour une classe donnée.
Simuler des collaborateurs dans un test permet et	n général de limiter le coût d'exécution d'un test.
Il existe des librairies pour créer des collaborate programmation courants.	urs simulés dans à peu près tous les langages de
On utilise des collaborateurs simulés en général d	dans les tests d'intégration.
Un collaborateur simulé est communément appel	é en anglais "mock object".
☐ Il n'est pas possible de créer des collaborateurs si	mulés dans des tests en PHP.
Question 3 & Code coverage - Cochez les assertion	ns vraies :
Dans un projet agile, l'objectif de couverture de définition de fini.	code par les tests est un bon critère à ajouter à la
La couverture du code par les tests permet de sa sont invoquées par les tests.	avoir quelles classes, méthodes et lignes de codes
En TDD, la couverture du code par les tests n'est	pas nécéssairement de 100%.
La couverture du code par les tests permet de me	surer objectivement la qualité des tests.

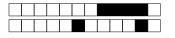




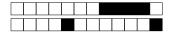
3 3 3 3 3 3 3 4 Veuillez coder votre numéro
d'étudiant ci-contre, et écrire votre nom
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
Nom et prénom :
9 9 9 9 9 9
Durée totale : 42 mn (56 mn pour les tiers temps). Partie QCM sur 7 points. Aucun document ni dispositif électronique n'est autorisé. Le symbole "trèfle" sur une question indique qu'il peut être attendu plus d'une réponse correcte. Les réponses doivent être fournies uniquement sur les feuilles d'énoncé. Aucune autre réponse ne sera prise en compte dans la notation. Les cases doivent être entièrement coloriées en noir pour être prises en compte dans la notation. Pour modifier une réponse, vous pouvez utiliser un correcteur blanc, sans re-dessiner la case à cocher.
Question 1 & Code coverage - Cochez les assertions vraies :
La couverture du code par les tests permet de mesurer objectivement la qualité des tests.
Dans un projet agile, l'objectif de couverture de code par les tests est un bon critère à ajouter à la définition de fini.
En TDD, la couverture du code par les tests n'est pas nécéssairement de 100%.
La couverture du code par les tests permet de savoir quelles classes, méthodes et lignes de codes sont invoquées par les tests.
Question 2 Répartition des tests - Dans une application, quelle répartition de quantités de tests est la plus pertinente :
80% de tests d'intégration, 15% de tests unitaires, 5% de tests UI.
80% de tests unitaires, 15% de tests d'intégration, 5% de tests UI.
80% de tests UI, 15% de tests d'intégration, 5% de tests unitaires.
Question 3 🌲 Tests unitaires vs tests d'intégration - Cochez les assertions vraies :
Les tests unitaires créent toujours les vrais collaborateurs pour tester un objet.
Les tests d'intégration s'exécutent en général plus rapidement que les tests unitaires.
Les tests d'intégration vérifient le fonctionnement de "vrais" objets en interactions.
Les tests unitaires consomment en général plus de ressources que les tests d'intégration.



Question 4 🌲 💮 S	Suite de tests - Cochez les assertions vraies :
Les tests des Une bonne s Un test en e	suite de tests teste explicitement uniquement l'interface publique de l'objet testé. s cas d'erreur suffisent pour tester la robustesse d'un programme. suite de tests est une suite de tests où les tests dépendent les uns des autres. rreur indique en général un bug dans le test. chec indique en général un bug dans le programme principal.
Question 5 🌲	Simuler des collaborateurs - Cochez les assertions vraies :
Simuler des Il n'est pas t Il n'est pas p Un collabora Il existe des	es collaborateurs simulés en général dans les tests d'intégration. collaborateurs dans un test permet en général de limiter le coût d'exécution d'un test. coujours possible de créer un collaborateur simulé "à la main" pour une classe donnée. cossible de créer des collaborateurs simulés dans des tests en PHP. ateur simulé est communément appelé en anglais "mock object". librairies pour créer des collaborateurs simulés dans à peu près tous les langages de tion courants.
Question 6 👫	TDD - Cochez les assertions vraies :
En TDD, on En TDD, dès Le refactoria	éliore la conception d'un programme. code tous les tests d'une classe avant de coder toutes les méthodes de la classe. s qu'un test passe, on passe à l'écriture d'un nouveau test. ng est une étape du TDD qu'un développeur doit entreprendre après qu'un test échoue. ng consiste à remanier le code d'un programme.
Question 7 🐥 🛚 I	Bénéfices et limites des tests automatisés (TA) - Cochez les assertions vraies :
Les TA pern Les TA peuv	nettent de s'assurer qu'un programme fonctionne comme attendu. nettent de limiter le nombre de bugs. vent contenir des bugs. nettent d'effectuer des changements dans le code en confiance.



\square_2 \square_2 \square_2 \square_2 \square_2 \square_2 \square_2	
	Veuillez coder votre numéro d'étudiant ci-contre, et écrire votre nom
	dans la case ci-dessous.
5 5 5 5 5 5 5	
6 6 6 6 6 6 6 6	Nom et prénom :
Durée totale : 42 mn (56 mn pour les tiers temps). Aucun document ni dispositif électronique n'est au Le symbole "trèfle" sur une question indique qu'il p Les réponses doivent être fournies uniquement sur prise en compte dans la notation. Les cases doivent ê en compte dans la notation. Pour modifier une répon re-dessiner la case à cocher.	peut être attendu plus d'une réponse correcte. les feuilles d'énoncé. Aucune autre réponse ne sera etre entièrement coloriées en noir pour être prises
Question 1 Tests unitaires vs tests d'intégration Les tests d'intégration vérifient le fonctionnement Les tests unitaires consomment en général plus de la test unitaires créent toujours les vrais collais Les tests d'intégration s'exécutent en général plus de la test de la test d'intégration s'exécutent en général plus de la test de la test d'intégration s'exécutent en général plus d'intégration s'exécutent en général plus de la test d'intégration s'exécutent en général plus de la test d'intégration s'exécutent en général plus d'intégration s'exécutent en général plus d'intégration s'exécutent en général plus d'intégration d'intégration s'exécutent en général plus d'intégration s'exécutent en général plus d'intégration d'intégration d'intégration s'exécutent en général plus d'intégration d'intégrati	nt de "vrais" objets en interactions. de ressources que les tests d'intégration. borateurs pour tester un objet.
Question 2 ♣ TDD - Cochez les assertions vraies :	
Le TDD améliore la conception d'un programme	
En TDD, on code tous les tests d'une classe avan	
En TDD, dès qu'un test passe, on passe à l'écritu	re d'un nouveau test.
Le refactoring est une étape du TDD qu'un dével	loppeur doit entreprendre après qu'un test échoue.
Le refactoring consiste à remanier le code d'un p	programme.
Question 3 🌲 Simuler des collaborateurs - Cochez	les assertions vraies :
☐ Il n'est pas possible de créer des collaborateurs s	imulés dans des tests en PHP.
Il existe des librairies pour créer des collaborate programmation courants.	eurs simulés dans à peu près tous les langages de
☐ Il n'est pas toujours possible de créer un collabo	rateur simulé "à la main" pour une classe donnée.
Un collaborateur simulé est communément appe	elé en anglais "mock object".
Simuler des collaborateurs dans un test permet e	en général de limiter le coût d'exécution d'un test.
On utilise des collaborateurs simulés en général	dans les tests d'intégration.



Question 4 - Suite de tests - Cochez les assertions vraies :
Une bonne suite de tests est une suite de tests où les tests dépendent les uns des autres.
Une bonne suite de tests teste explicitement uniquement l'interface publique de l'objet testé.
Un test en erreur indique en général un bug dans le test.
Un test en échec indique en général un bug dans le programme principal.
Les tests des cas d'erreur suffisent pour tester la robustesse d'un programme.
Question 5 & Bénéfices et limites des tests automatisés (TA) - Cochez les assertions vraies :
Les TA permettent de limiter le nombre de bugs.
Les TA peuvent contenir des bugs.
Les TA permettent d'effectuer des changements dans le code en confiance.
Les TA permettent de s'assurer qu'un programme fonctionne comme attendu.
Question 6 Répartition des tests - Dans une application, quelle répartition de quantités de tests est la plus pertinente :
80% de tests UI, 15% de tests d'intégration, 5% de tests unitaires.
80% de tests d'intégration, 15% de tests unitaires, 5% de tests UI.
80% de tests unitaires, 15% de tests d'intégration, 5% de tests UI.
Question 7 - Code coverage - Cochez les assertions vraies :
En TDD, la couverture du code par les tests n'est pas nécéssairement de 100%.
Dans un projet agile, l'objectif de couverture de code par les tests est un bon critère à ajouter à la définition de fini.
La couverture du code par les tests permet de savoir quelles classes, méthodes et lignes de codes sont invoquées par les tests.
La couverture du code par les tests permet de mesurer objectivement la qualité des tests.

	0
d'étudiant ci-contre, et écrire votre nor	n
444444 dans la case ci-dessous.	
Durée totale : 42 mn (56 mn pour les tiers temps). Partie QCM sur 7 points.	
Aucun document ni dispositif électronique n'est autorisé.	
Le symbole "trèfle" sur une question indique qu'il peut être attendu plus d'une réponse correc Les réponses doivent être fournies uniquement sur les feuilles d'énoncé. Aucune autre réponse	
prise en compte dans la notation. Les cases doivent être entièrement coloriées en noir pour êtr	
en compte dans la notation. Pour modifier une réponse, vous pouvez utiliser un correcteur bla	nc, sans
re-dessiner la case à cocher.	
Question 1 - Suite de tests - Cochez les assertions vraies :	
Une bonne suite de tests teste explicitement uniquement l'interface publique de l'objet test	é.
Un test en échec indique en général un bug dans le programme principal.	
Un test en erreur indique en général un bug dans le test.	
Une bonne suite de tests est une suite de tests où les tests dépendent les uns des autres.	
Les tests des cas d'erreur suffisent pour tester la robustesse d'un programme.	
Question 2 - Cochez les assertions vraies :	
Les tests d'intégration vérifient le fonctionnement de "vrais" objets en interactions.	
Les tests unitaires créent toujours les vrais collaborateurs pour tester un objet.	
Les tests unitaires consomment en général plus de ressources que les tests d'intégration.	
Les tests d'intégration s'exécutent en général plus rapidement que les tests unitaires.	
Question 3 Répartition des tests - Dans une application, quelle répartition de quantités de la plus pertinente :	tests est
80% de tests unitaires, 15% de tests d'intégration, 5% de tests UI.	
80% de tests d'intégration, 15% de tests unitaires, 5% de tests UI.	
80% de tests UI, 15% de tests d'intégration, 5% de tests unitaires.	
Question 4 - Bénéfices et limites des tests automatisés (TA) - Cochez les assertions vraies :	
Les TA permettent d'effectuer des changements dans le code en confiance.	
Les TA permettent de s'assurer qu'un programme fonctionne comme attendu.	
Les TA permettent de limiter le nombre de bugs.	
Les TA peuvent contenir des bugs.	

Question 5 - Simuler des collaborateurs - Cochez les assertions vraies :
Il n'est pas toujours possible de créer un collaborateur simulé "à la main" pour une classe donnée.
Il existe des librairies pour créer des collaborateurs simulés dans à peu près tous les langages de programmation courants.
On utilise des collaborateurs simulés en général dans les tests d'intégration.
Un collaborateur simulé est communément appelé en anglais "mock object".
Il n'est pas possible de créer des collaborateurs simulés dans des tests en PHP.
Simuler des collaborateurs dans un test permet en général de limiter le coût d'exécution d'un test.
Question 6 A TDD - Cochez les assertions vraies :
En TDD, dès qu'un test passe, on passe à l'écriture d'un nouveau test.
Le refactoring consiste à remanier le code d'un programme.
Le TDD améliore la conception d'un programme.
Le refactoring est une étape du TDD qu'un développeur doit entreprendre après qu'un test échoue
En TDD, on code tous les tests d'une classe avant de coder toutes les méthodes de la classe.
Question 7 & Code coverage - Cochez les assertions vraies :
Dans un projet agile, l'objectif de couverture de code par les tests est un bon critère à ajouter à la définition de fini.
En TDD, la couverture du code par les tests n'est pas nécéssairement de 100%.
La couverture du code par les tests permet de mesurer objectivement la qualité des tests.
La couverture du code par les tests permet de savoir quelles classes, méthodes et lignes de codes sont invoquées par les tests.

2023-2024 - R4-02 Qualité de développement - CC1 $\boxed{}$ $\boxed{}$

Veuillez coder votre numéro d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous.

• •

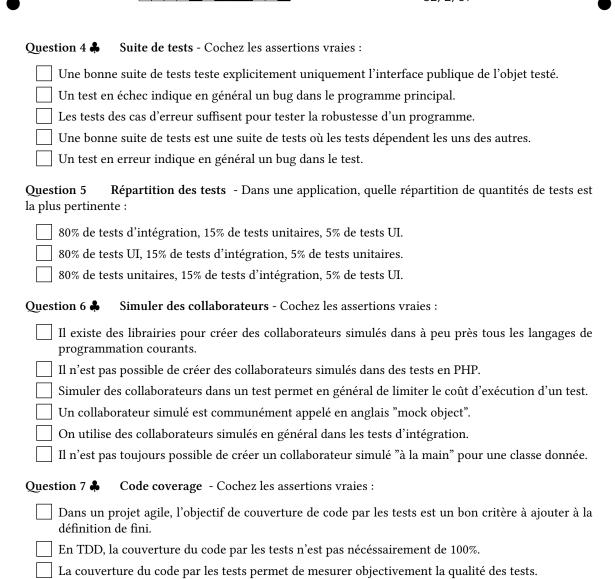
Durée totale : 42 mn (56 mn pour les tiers temps). Partie QCM sur 7 points.

Aucun document ni dispositif électronique n'est autorisé.

Le symbole "trèfle" sur une question indique qu'il peut être attendu plus d'une réponse correcte.

Les réponses doivent être fournies uniquement sur les feuilles d'énoncé. Aucune autre réponse ne sera prise en compte dans la notation. Les cases doivent être entièrement coloriées en noir pour être prises en compte dans la notation. Pour modifier une réponse, vous pouvez utiliser un correcteur blanc, sans

re-dessiner la case à cocher.	
Question 1 ૈ	Tests unitaires vs tests d'intégration - Cochez les assertions vraies :
Les tests u	unitaires créent toujours les vrais collaborateurs pour tester un objet.
Les tests o	l'intégration s'exécutent en général plus rapidement que les tests unitaires.
Les tests ı	initaires consomment en général plus de ressources que les tests d'intégration.
Les tests o	l'intégration vérifient le fonctionnement de "vrais" objets en interactions.
Question 2 ૈ	TDD - Cochez les assertions vraies :
Le TDD a	méliore la conception d'un programme.
Le refacto	ring est une étape du TDD qu'un développeur doit entreprendre après qu'un test échoue.
Le refacto	ring consiste à remanier le code d'un programme.
En TDD, o	lès qu'un test passe, on passe à l'écriture d'un nouveau test.
En TDD, o	on code tous les tests d'une classe avant de coder toutes les méthodes de la classe.
Question 3 🌲	Bénéfices et limites des tests automatisés (TA) - Cochez les assertions vraies :
Les TA pe	rmettent de limiter le nombre de bugs.
Les TA pe	uvent contenir des bugs.
Les TA pe	rmettent de s'assurer qu'un programme fonctionne comme attendu.
Les TA pe	rmettent d'effectuer des changements dans le code en confiance.



La couverture du code par les tests permet de savoir quelles classes, méthodes et lignes de codes

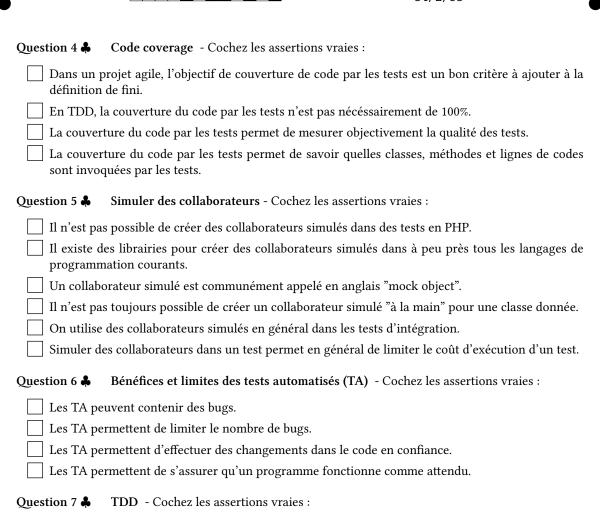
sont invoquées par les tests.

0 0	
Durée totale : 42 mn (56 mn pour les tiers temps) Aucun document ni dispositif électronique n'est a Le symbole "trèfle" sur une question indique qu'il Les réponses doivent être fournies uniquement su prise en compte dans la notation. Les cases doivent en compte dans la notation. Pour modifier une réporre-dessiner la case à cocher.	autorisé. l peut être attendu plus d'une réponse correcte. ur les feuilles d'énoncé. Aucune autre réponse ne sera c être entièrement coloriées en noir pour être prises
Question 1 ♣ TDD - Cochez les assertions vraies Le refactoring est une étape du TDD qu'un dév Le refactoring consiste à remanier le code d'un En TDD, dès qu'un test passe, on passe à l'écrit En TDD, on code tous les tests d'une classe ava Le TDD améliore la conception d'un programm	veloppeur doit entreprendre après qu'un test échoue. n programme. ture d'un nouveau test. ant de coder toutes les méthodes de la classe.
Question 2 ♣ Suite de tests - Cochez les assertion Une bonne suite de tests est une suite de tests en général un bug dar Les tests des cas d'erreur suffisent pour tester l Un test en erreur indique en général un bug dar Une bonne suite de tests teste explicitement un	où les tests dépendent les uns des autres. ns le programme principal. la robustesse d'un programme. ans le test.
Un collaborateur simulé est communément app Il existe des librairies pour créer des collabora	t en général de limiter le coût d'exécution d'un test.
programmation courants. On utilise des collaborateurs simulés en généra Il n'est pas toujours possible de créer un collab Il n'est pas possible de créer des collaborateurs	porateur simulé "à la main" pour une classe donnée.

Question 4 ૈ	Tests unitaires vs tests d'intégration - Cochez les assertions vraies :
Les tests	d'intégration s'exécutent en général plus rapidement que les tests unitaires.
Les tests	unitaires créent toujours les vrais collaborateurs pour tester un objet.
Les tests	d'intégration vérifient le fonctionnement de "vrais" objets en interactions.
Les tests	unitaires consomment en général plus de ressources que les tests d'intégration.
Question 5 ૈ	Code coverage - Cochez les assertions vraies :
	rture du code par les tests permet de savoir quelles classes, méthodes et lignes de codes quées par les tests.
La couver	ture du code par les tests permet de mesurer objectivement la qualité des tests.
En TDD,	la couverture du code par les tests n'est pas nécéssairement de 100%.
Dans un j définition	projet agile, l'objectif de couverture de code par les tests est un bon critère à ajouter à la 1 de fini.
Question 6 la plus pertinen	Répartition des tests - Dans une application, quelle répartition de quantités de tests est te :
80% de te	sts unitaires, 15% de tests d'intégration, 5% de tests UI.
80% de te	sts d'intégration, 15% de tests unitaires, 5% de tests UI.
80% de te	sts UI, 15% de tests d'intégration, 5% de tests unitaires.
Question 7 🌲	Bénéfices et limites des tests automatisés (TA) - Cochez les assertions vraies :
Les TA pe	ermettent de s'assurer qu'un programme fonctionne comme attendu.
Les TA pe	euvent contenir des bugs.
Les TA pe	ermettent d'effectuer des changements dans le code en confiance.
Les TA pe	ermettent de limiter le nombre de bugs.



0 0	← Veuillez coder votre numéro d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous. Nom et prénom :
Durée totale : 42 mn (56 mn pour les tiers temps). F Aucun document ni dispositif électronique n'est au Le symbole "trèfle" sur une question indique qu'il p Les réponses doivent être fournies uniquement sur le prise en compte dans la notation. Les cases doivent êt en compte dans la notation. Pour modifier une répons re-dessiner la case à cocher.	torisé. eut être attendu plus d'une réponse correcte. es feuilles d'énoncé. Aucune autre réponse ne sera re entièrement coloriées en noir pour être prises
Question 1 ♣ Suite de tests - Cochez les assertions : Une bonne suite de tests est une suite de tests où Un test en erreur indique en général un bug dans Un test en échec indique en général un bug dans Les tests des cas d'erreur suffisent pour tester la r Une bonne suite de tests teste explicitement unique	les tests dépendent les uns des autres. le test. le programme principal. cobustesse d'un programme.
Question 2 Répartition des tests - Dans une appli la plus pertinente : 80% de tests unitaires, 15% de tests d'intégration, 5% de la 180% de tests UI, 15% de tests d'intégration, 5% de la 80% de tests d'intégration, 15% de tests unitaires,	tests unitaires.
Question 3 Tests unitaires vs tests d'intégration Les tests unitaires consomment en général plus d Les tests unitaires créent toujours les vrais collab Les tests d'intégration vérifient le fonctionnement Les tests d'intégration s'exécutent en général plu	e ressources que les tests d'intégration. orateurs pour tester un objet. It de "vrais" objets en interactions.



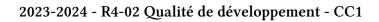
Le refactoring consiste à remanier le code d'un programme.

Le TDD améliore la conception d'un programme.

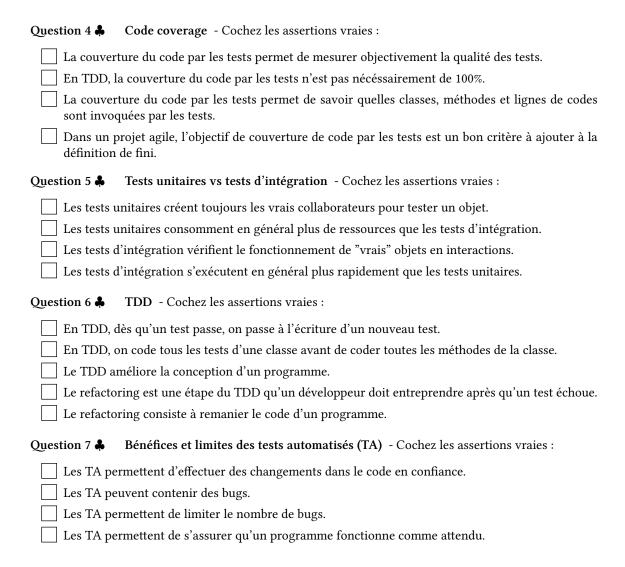
En TDD, dès qu'un test passe, on passe à l'écriture d'un nouveau test.

En TDD, on code tous les tests d'une classe avant de coder toutes les méthodes de la classe.

Le refactoring est une étape du TDD qu'un développeur doit entreprendre après qu'un test échoue.



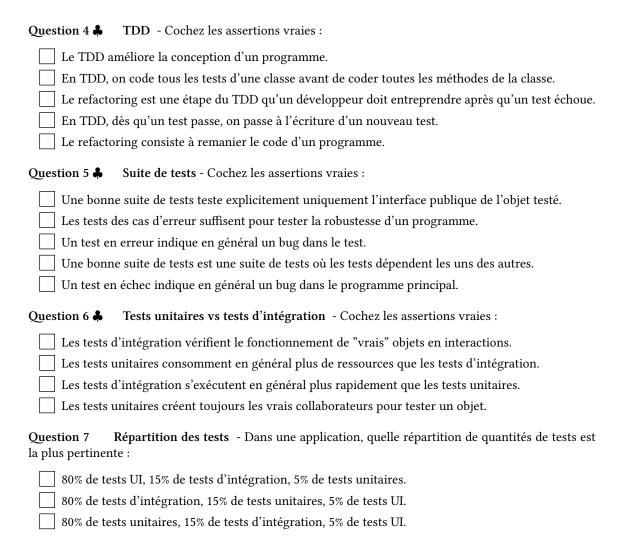
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	
	← Veuillez coder votre numéro
	d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous.
	Nom et prénom :
Durée totale : 42 mn (56 mn pour les tiers temps). Pa	
Aucun document ni dispositif électronique n'est auto	
Le symbole "trèfle" sur une question indique qu'il pe Les réponses doivent être fournies uniquement sur le	
prise en compte dans la notation. Les cases doivent êtr	
en compte dans la notation. Pour modifier une réponse	e, vous pouvez utiliser un correcteur blanc, sans
re-dessiner la case à cocher.	
Question 1 & Suite de tests - Cochez les assertions v	rraies :
Un test en échec indique en général un bug dans l	e programme principal
Une bonne suite de tests teste explicitement uniqu	
Un test en erreur indique en général un bug dans	
Les tests des cas d'erreur suffisent pour tester la re	
Une bonne suite de tests est une suite de tests où l	
	-
Question 2 - Cochez le	es assertions vraies :
Il n'est pas possible de créer des collaborateurs sir	nulés dans des tests en PHP.
Il existe des librairies pour créer des collaborateu programmation courants.	urs simulés dans à peu près tous les langages de
Simuler des collaborateurs dans un test permet en	général de limiter le coût d'exécution d'un test.
Un collaborateur simulé est communément appelé	é en anglais "mock object".
☐ Il n'est pas toujours possible de créer un collabora	teur simulé "à la main" pour une classe donnée.
On utilise des collaborateurs simulés en général d	ans les tests d'intégration.
Question 3 Répartition des tests - Dans une applic la plus pertinente :	cation, quelle répartition de quantités de tests est
80% de tests d'intégration, 15% de tests unitaires, 5	5% de tests UI.
80% de tests unitaires, 15% de tests d'intégration, 5	
80% de tests UI, 15% de tests d'intégration, 5% de t	



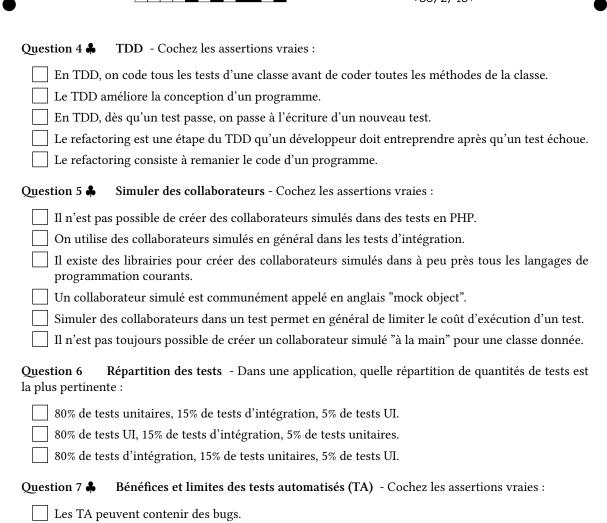
\square_2 \square_2 \square_2 \square_2 \square_2 \square_2 \square_2	
$\square 3 \square 3$	← Veuillez coder votre numéro
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous.
	Nom et prénom :
Durée totale : 42 mn (56 mn pour les tiers temps). I Aucun document ni dispositif électronique n'est au Le symbole "trèfle" sur une question indique qu'il p Les réponses doivent être fournies uniquement sur l prise en compte dans la notation. Les cases doivent ê en compte dans la notation. Pour modifier une répon re-dessiner la case à cocher.	torisé. Deut être attendu plus d'une réponse correcte. les feuilles d'énoncé. Aucune autre réponse ne sera tre entièrement coloriées en noir pour être prises
Question 1 - Code coverage - Cochez les assertion	ns vraies :
En TDD, la couverture du code par les tests n'est	pas nécéssairement de 100%.
La couverture du code par les tests permet de mesurer objectivement la qualité des tests.	
Dans un projet agile, l'objectif de couverture de code par les tests est un bon critère à ajouter à la définition de fini.	
La couverture du code par les tests permet de sa sont invoquées par les tests.	avoir quelles classes, méthodes et lignes de codes
Question 2 Répartition des tests - Dans une appli la plus pertinente :	ication, quelle répartition de quantités de tests est
80% de tests UI, 15% de tests d'intégration, 5% de	tests unitaires.
80% de tests d'intégration, 15% de tests unitaires,	, 5% de tests UI.
80% de tests unitaires, 15% de tests d'intégration,	, 5% de tests UI.
Question 3 ♣ TDD - Cochez les assertions vraies :	
Le refactoring est une étape du TDD qu'un dével	oppeur doit entreprendre après qu'un test échoue.
Le TDD améliore la conception d'un programme	
Le refactoring consiste à remanier le code d'un p	orogramme.
En TDD, on code tous les tests d'une classe avan	t de coder toutes les méthodes de la classe.
En TDD, dès qu'un test passe, on passe à l'écritu	re d'un nouveau test.

Questi	on 4 🌲 Tests unitaires vs tests d'intégration - Cochez les assertions vraies :
I	es tests unitaires consomment en général plus de ressources que les tests d'intégration.
	les tests d'intégration vérifient le fonctionnement de "vrais" objets en interactions.
	es tests unitaires créent toujours les vrais collaborateurs pour tester un objet.
	es tests d'intégration s'exécutent en général plus rapidement que les tests unitaires.
Questi	on 5 - Simuler des collaborateurs - Cochez les assertions vraies :
	Simuler des collaborateurs dans un test permet en général de limiter le coût d'exécution d'un test.
	l existe des librairies pour créer des collaborateurs simulés dans à peu près tous les langages de programmation courants.
I	l n'est pas possible de créer des collaborateurs simulés dans des tests en PHP.
J	In collaborateur simulé est communément appelé en anglais "mock object".
	On utilise des collaborateurs simulés en général dans les tests d'intégration.
I	l n'est pas toujours possible de créer un collaborateur simulé "à la main" pour une classe donnée.
Questi	on 6 - Suite de tests - Cochez les assertions vraies :
J	Une bonne suite de tests est une suite de tests où les tests dépendent les uns des autres.
I	es tests des cas d'erreur suffisent pour tester la robustesse d'un programme.
J	In test en erreur indique en général un bug dans le test.
J	Une bonne suite de tests teste explicitement uniquement l'interface publique de l'objet testé.
J	In test en échec indique en général un bug dans le programme principal.
Questi	on 7 🌲 Bénéfices et limites des tests automatisés (TA) - Cochez les assertions vraies :
	es TA peuvent contenir des bugs.
	es TA permettent de limiter le nombre de bugs.
	es TA permettent de s'assurer qu'un programme fonctionne comme attendu.
П	es TA permettent d'effectuer des changements dans le code en confiance.

0 0	← Veuillez coder votre numéro d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous. Nom et prénom:
Durée totale : 42 mn (56 mn pour les tiers temps). Pa Aucun document ni dispositif électronique n'est auto Le symbole "trèfle" sur une question indique qu'il pe Les réponses doivent être fournies uniquement sur le prise en compte dans la notation. Les cases doivent êtr en compte dans la notation. Pour modifier une réponse re-dessiner la case à cocher.	orisé. ut être attendu plus d'une réponse correcte. s feuilles d'énoncé. Aucune autre réponse ne sera e entièrement coloriées en noir pour être prises
Question 1 ♣ Bénéfices et limites des tests automat Les TA permettent de s'assurer qu'un programme Les TA permettent d'effectuer des changements de Les TA peuvent contenir des bugs. Les TA permettent de limiter le nombre de bugs.	
Question 2 . Code coverage - Cochez les assertions	s vraies :
Dans un projet agile, l'objectif de couverture de c définition de fini.	ode par les tests est un bon critère à ajouter à la
La couverture du code par les tests permet de sav sont invoquées par les tests.	voir quelles classes, méthodes et lignes de codes
La couverture du code par les tests permet de mes	urer objectivement la qualité des tests.
En TDD, la couverture du code par les tests n'est p	oas nécéssairement de 100%.
Question 3 . Simuler des collaborateurs - Cochez le	es assertions vraies :
Il n'est pas possible de créer des collaborateurs sin	nulés dans des tests en PHP.
On utilise des collaborateurs simulés en général d	ans les tests d'intégration.
Un collaborateur simulé est communément appelé	en anglais "mock object".
Il existe des librairies pour créer des collaborateur programmation courants.	rs simulés dans à peu près tous les langages de
Simuler des collaborateurs dans un test permet en	général de limiter le coût d'exécution d'un test.
Il n'est pas toujours possible de créer un collabora	teur simulé "à la main" pour une classe donnée.



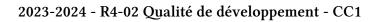
\square_2 \square_2 \square_2 \square_2 \square_2 \square_2 \square_2 \square_2	
	← Veuillez coder votre numéro
	d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous.
	dans in case of desisons.
	Nom et prénom :
Durée totale : 42 mn (56 mn pour les tiers temps). Pa Aucun document ni dispositif électronique n'est auto Le symbole "trèfle" sur une question indique qu'il pe Les réponses doivent être fournies uniquement sur les prise en compte dans la notation. Les cases doivent être en compte dans la notation. Pour modifier une réponse re-dessiner la case à cocher.	orisé. ut être attendu plus d'une réponse correcte. s feuilles d'énoncé. Aucune autre réponse ne sera e entièrement coloriées en noir pour être prises
Question 1 & Code coverage - Cochez les assertions	vraies :
La couverture du code par les tests permet de mes	urer objectivement la qualité des tests.
En TDD, la couverture du code par les tests n'est p	
La couverture du code par les tests permet de sav sont invoquées par les tests.	
Dans un projet agile, l'objectif de couverture de condéfinition de fini.	ode par les tests est un bon critère à ajouter à la
Question 2 4 Tests unitaires vs tests d'intégration	- Cochez les assertions vraies :
Les tests unitaires créent toujours les vrais collabo	rateurs pour tester un objet.
Les tests d'intégration vérifient le fonctionnement	de "vrais" objets en interactions.
Les tests d'intégration s'exécutent en général plus	rapidement que les tests unitaires.
Les tests unitaires consomment en général plus de	ressources que les tests d'intégration.
Question 3 - Suite de tests - Cochez les assertions v	raies :
Les tests des cas d'erreur suffisent pour tester la ro	bustesse d'un programme.
Une bonne suite de tests teste explicitement uniqu	ement l'interface publique de l'objet testé.
Un test en échec indique en général un bug dans le	e programme principal.
Une bonne suite de tests est une suite de tests où l	es tests dépendent les uns des autres.
Un test en erreur indique en général un bug dans l	e test.



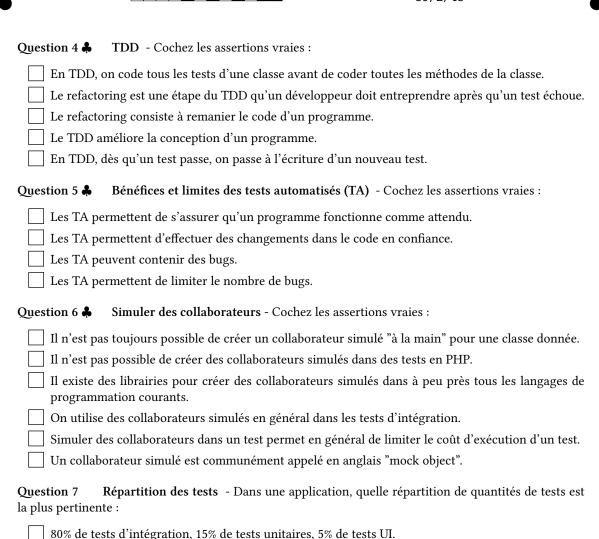
Les TA permettent de s'assurer qu'un programme fonctionne comme attendu.

Les TA permettent d'effectuer des changements dans le code en confiance.

Les TA permettent de limiter le nombre de bugs.



	← Veuillez coder votre numéro
	d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous.
	ualis la case ci-uessous.
	Nom et prénom :
Durée totale : 42 mn (56 mn pour les tiers temps). P Aucun document ni dispositif électronique n'est aut Le symbole "trèfle" sur une question indique qu'il p Les réponses doivent être fournies uniquement sur le prise en compte dans la notation. Les cases doivent êt en compte dans la notation. Pour modifier une répons re-dessiner la case à cocher.	corisé. eut être attendu plus d'une réponse correcte. es feuilles d'énoncé. Aucune autre réponse ne sera re entièrement coloriées en noir pour être prises
Question 1 - Suite de tests - Cochez les assertions	vraies :
Une bonne suite de tests teste explicitement uniq	uement l'interface publique de l'objet testé.
Une bonne suite de tests est une suite de tests où	les tests dépendent les uns des autres.
Un test en erreur indique en général un bug dans	le test.
Un test en échec indique en général un bug dans	le programme principal.
Les tests des cas d'erreur suffisent pour tester la r	obustesse d'un programme.
Question 2 🌲 Tests unitaires vs tests d'intégration	- Cochez les assertions vraies :
Les tests d'intégration vérifient le fonctionnemen	t de "vrais" objets en interactions.
Les tests unitaires consomment en général plus d	e ressources que les tests d'intégration.
Les tests unitaires créent toujours les vrais collab	orateurs pour tester un objet.
Les tests d'intégration s'exécutent en général plus	s rapidement que les tests unitaires.
Oraști a 2 • Orde arran Cada la constitu	
Question 3 & Code coverage - Cochez les assertion	is vraies :
Dans un projet agile, l'objectif de couverture de définition de fini.	code par les tests est un bon critère à ajouter à la
La couverture du code par les tests permet de sa sont invoquées par les tests.	voir quelles classes, méthodes et lignes de codes
☐ En TDD, la couverture du code par les tests n'est	pas nécéssairement de 100%.
La couverture du code par les tests permet de me	surer objectivement la qualité des tests.



80% de tests UI, 15% de tests d'intégration, 5% de tests unitaires. 80% de tests unitaires, 15% de tests d'intégration, 5% de tests UI.



$\begin{array}{cccccccccccccccccccccccccccccccccccc$	
	← Veuillez coder votre numéro
	d'étudiant ci-contre, et écrire votre nom dans la case ci-dessous.
	uans la case el dessous.
	Nom et prénom :
Durée totale : 42 mn (56 mn pour les tiers temps) Aucun document ni dispositif électronique n'est a Le symbole "trèfle" sur une question indique qu'il Les réponses doivent être fournies uniquement su prise en compte dans la notation. Les cases doivent en compte dans la notation. Pour modifier une réporre-dessiner la case à cocher.	autorisé. I peut être attendu plus d'une réponse correcte. r les feuilles d'énoncé. Aucune autre réponse ne sera être entièrement coloriées en noir pour être prises
Question 1 Simuler des collaborateurs - Coche	ez les assertions vraies : t en général de limiter le coût d'exécution d'un test.
Il n'est pas possible de créer des collaborateurs	
On utilise des collaborateurs simulés en généra	
	ateurs simulés dans à peu près tous les langages de
Un collaborateur simulé est communément app	pelé en anglais "mock object".
Il n'est pas toujours possible de créer un collab	orateur simulé "à la main" pour une classe donnée.
Question 2 🌲 Bénéfices et limites des tests auton	natisés (TA) - Cochez les assertions vraies :
Les TA peuvent contenir des bugs.	
Les TA permettent d'effectuer des changement	s dans le code en confiance.
Les TA permettent de s'assurer qu'un program	me fonctionne comme attendu.
Les TA permettent de limiter le nombre de bug	S.
Question 3 - Suite de tests - Cochez les assertion	ns vraies :
Les tests des cas d'erreur suffisent pour tester l	a robustesse d'un programme.
Un test en échec indique en général un bug da	ns le programme principal.
Un test en erreur indique en général un bug da	ns le test.
Une bonne suite de tests est une suite de tests o	où les tests dépendent les uns des autres.
Une bonne suite de tests teste explicitement un	niquement l'interface publique de l'objet testé.



Question 4 ♣ TDD - Cochez les assertions vraies :
En TDD, dès qu'un test passe, on passe à l'écriture d'un nouveau test.
Le TDD améliore la conception d'un programme.
Le refactoring consiste à remanier le code d'un programme.
Le refactoring est une étape du TDD qu'un développeur doit entreprendre après qu'un test échoue.
En TDD, on code tous les tests d'une classe avant de coder toutes les méthodes de la classe.
Question 5 Répartition des tests - Dans une application, quelle répartition de quantités de tests est la plus pertinente :
80% de tests d'intégration, 15% de tests unitaires, 5% de tests UI.
80% de tests unitaires, 15% de tests d'intégration, 5% de tests UI.
80% de tests UI, 15% de tests d'intégration, 5% de tests unitaires.
Question 6 - Tests unitaires vs tests d'intégration - Cochez les assertions vraies :
Les tests unitaires consomment en général plus de ressources que les tests d'intégration.
Les tests d'intégration vérifient le fonctionnement de "vrais" objets en interactions.
Les tests d'intégration s'exécutent en général plus rapidement que les tests unitaires.
Les tests unitaires créent toujours les vrais collaborateurs pour tester un objet.
Question 7 ♣ Code coverage - Cochez les assertions vraies :
La couverture du code par les tests permet de mesurer objectivement la qualité des tests.
La couverture du code par les tests permet de savoir quelles classes, méthodes et lignes de codes sont invoquées par les tests.
En TDD, la couverture du code par les tests n'est pas nécéssairement de 100%.
Dans un projet agile, l'objectif de couverture de code par les tests est un bon critère à ajouter à la définition de fini.