dans la nuit

fl (https://profile.intra.42.fr/searches)

(https://profile.intra.42.fr)

ÉCHELLE DU PROJET CPP MODULE 05 (/PROJETS/CPP-MODULE-05)

Vous devez évaluer 1 étudiant dans cette équipe

Dépôt Git

git@vogsphere.msk.21-school.ru:vogsphere/intra-uuid-193aaa9

Introduction

Veuillez respecter les règles suivantes :

- Restez poli, courtois, respectueux et constructif tout au long du processus d'évaluation. Le bien-être de la communauté en dépend.
- Identifier avec l'étudiant ou le groupe dont le travail est évalué les dysfonctionnements possibles de son projet. Prendre le temps de discuter et de débattre des problèmes éventuellement identifiés.
- Il est important de prendre en compte les différences de compréhension entre vos pairs quant aux instructions du projet et à l'étendue de ses fonctionnalités.
 Gardez toujours l'esprit ouvert et notez-les aussi honnêtement que possible. La pédagogie n'est utile que si l'évaluation par les pairs est menée avec sérieux.

Lignes directrices

- Notez uniquement le travail qui a été rendu dans le dépôt Git de l'étudiant ou du groupe évalué.
- Vérifiez que le dépôt Git appartient bien à l'étudiant(e). Assurez-vous que le projet est bien celui attendu. Vérifiez également que « git clone » est utilisé dans un dossier vide.
- Vérifiez soigneusement qu'aucun alias malveillant n'a été utilisé pour vous tromper et vous faire évaluer quelque chose qui n'est pas le contenu du dépôt officiel.
- Pour éviter toute surprise et le cas échéant, revoir ensemble les scripts utilisés

29/01/22, 19h30

pour faciliter la notation (scripts de test ou d'automatisation).

- Si vous n'avez pas terminé le devoir que vous allez évaluer, vous devez lire l'intégralité du sujet avant de commencer le processus d'évaluation.
- Utilisez les indicateurs disponibles pour signaler un référentiel vide, un programme non fonctionnel, une erreur de norme, une tricherie, etc.

Dans ces cas, le processus d'évaluation est terminé et la note finale est de 0, ou -42 en cas de tricherie. Cependant, sauf en cas de tricherie, il est fortement recommandé aux étudiants de relire ensemble les travaux rendus afin d'identifier les erreurs à éviter.

- Vous ne devriez jamais avoir à modifier un fichier, sauf le fichier de configuration, s'il existe.
 Si vous souhaitez modifier un fichier, prenez le temps d'en expliquer les raisons à
 l'étudiant évalué et assurez-vous que vous êtes tous les deux d'accord.
- Il est également nécessaire de vérifier l'absence de fuites mémoire. Toute mémoire allouée sur le tas doit être correctement libérée avant la fin de l'exécution.

Vous êtes autorisé à utiliser tous les outils disponibles sur votre ordinateur, tels que leaks, valgrind ou e_fence. En cas de fuite de mémoire, cochez l'option appropriée.

Pièces jointes

sujet.pdf (https://cdn.intra.42.fr/pdf/pdf/40215/sujet.pdf)

Tests préliminaires

En cas de suspicion de triche, l'évaluation s'arrête ici. Utilisez l'indicateur « Triche » pour le signaler. Prenez cette décision avec calme et prudence, et utilisez ce bouton avec prudence.

Prérequis

Le code doit être compilé avec C++ et les options -Wall -Wextra -Werror. N'oubliez pas que ce projet doit respecter la norme C++98. Par conséquent, les fonctions ou conteneurs C++11 (et versions ultérieures) ne sont PAS attendus.

L'un de ces éléments signifie que vous ne devez pas noter l'exercice en question :

- Une fonction est implémentée dans un fichier d'en-tête (à l'exception des fonctions de modèle).
- Un Makefile se compile sans les drapeaux requis et/ou un autre compilateur que c++.

Chacun de ces éléments signifie que vous devez marquer le projet avec « Interdit ». Fonction":

- Utilisation d'une fonction "C" (*alloc, *printf, free).
- Utilisation d'une fonction non autorisée dans les consignes d'exercice.

- Utilisation de « using namespace » ou le mot-clé « ami ».
- Utilisation d'une bibliothèque externe ou de fonctionnalités de versions autres que

Oui Non

Ex00 : Maman, quand je serai grand, je veux être bureaucrate !

Comme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le cas contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.

Bureaucrate

Il existe un Makefile qui se compile en utilisant les indicateurs appropriés.

Il existe une classe de bureaucrates. Son nom est constant.

Sa note varie de 1 (la plus élevée) à 150 (la plus basse).

Des exceptions sont levées lorsque vous essayez de créer un bureaucrate avec une note trop

élevée ou trop basse.

Il existe des accesseurs pour les attributs.

Il existe des fonctions permettant d'incrémenter ou de décrémenter la note.

Elles génèrent des exceptions lorsque cela est approprié. N'oubliez pas qu'incrémenter une note de 3

donne la note de 2 (1 étant la note la plus élevée).

Les exceptions utilisées héritent de std::exception, ou de quelque chose dérivé de

std::exception (c'est-à-dire qu'elles sont capturables comme std::exception & e).

Il existe un opérateur << pour la surcharge ostream qui renvoie les informations du Bureaucrate.

Oui

Non

Ex01: Formez-vous, asticots!

Comme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le cas contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas. cet exercice.

Formulaire

Il existe un Makefile qui se compile en utilisant les indicateurs appropriés.

Il existe une classe Form.

Il a un nom, un booléen qui indique s'il est signé (au début ce n'est pas le cas), une

note requise pour le signer et une note requise

pour l'exécuter.

es notes des formulaires suivent les mêmes contraintes que celles du Bureaucrate exceptions, 1 = le plus évent, 4.50 = le plus has, et ainsi de suited. seciate des accesseurs pour les attributs et un opérateur << pour la surcharge atteam qui affiche l'acts complet du formulaire. seciate une fonction membre Form:beSigned() qui fonctionne comme décrit par le sujet. Oui Non Non EXO2: Non, vous avez besoin du formulaire 28B, pas du 28C Oui Non Exo3: Non, vous avez besoin du formulaire 28B, pas du 28C omme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le sa contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice. se formulaires qui font réellement quelque chose existe du Makefille qui se compile en utilisant les indicateurs appropriés. sexiste des formes concrétes qui sont conformes aux spécifications de la matière (notes quisses, nome et actions). In expenent qu'un seul paramète dans leur constructeur, qui est leur cible. sexiste un Bureaucrat: execute/Bureaucrat const & executor) qui fonctionne comme prévu. Dans le suite méthode Form: execute/Bureaucrat const & executor) qui fonctionne comme spécifié par le sujet. Oui Non EXO3: Au moins, c'est mieux que de faire du café comme d'habitude, la fonction principale did contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le as contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxa, ne notez pas cet exercice.	Les noms et les grades sont constants.			
existe des accesseurs pour les attribute et un opérateur « pour la surcharge stream qui affiche l'état complet du formulaire existe une fonction membre Form:beSigned() qui fonctionne comme décrit par le sujet. existe une fonction Bureaucrat:signiForm() qui fonctionne comme décrit par le sujet. Oui Non Non Ex02: Non, vous avez besoin du formulaire 28B, pas du 28C Oui Non Ex03: Non, vous avez besoin du formulaire 28B, pas du 28C omme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prèvu. Dans le sa contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice. es formulaires qui font réellement quelque chose existe un Malterfile qui se compile en utilisant les indicateurs appropriés. existe des formes concrétés qui sont conformes aux spécifications de la matière (notes quius eu paramètre dans leur constructiour, qui est leur cible, existe une méthode Form:sexoute(Bursaucrat const & executor) qui fonctionne comme solicific par le sujet. Oui Non Ex03: Au moins, c'est mieux que de faire du café onte la classe dérivée qui exécute uniquement l'action. Se deux techniques sont valables. et deux techniques sont valables. Oui Non Ex03: Au moins, c'est mieux que de faire du café comme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le se contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.	Tous ces attributs sont privés et non protégés.			
existe une fonction membre Form::beSigned() qui fonctionne comme décrit par le sujet. Exx02 : Non, vous avez besoin du formulaire. Exx02 : Non, vous avez besoin du formulaire 28B, pas du 28C Oui Non Exx02 : Non, vous avez besoin du formulaire 28B, pas du 28C omme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le sa contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice. Si une classe un Makefille qui se compile en utilisant les indicateurs appropriés. existe un Makefille qui se compile en utilisant les indicateurs appropriés. existe une méthode Form: execute(Bureaucrat const & executor) qui fonctionne comme secficip par le sujet. il cette méthode est pure et les vérifications de qualité sont implementées dans chaque pue classe, soit cette méthode effectue les vérifications, puis appetle une autre méthode are la une des deux techniques sont vialubles. existe un Bureaucrat: execute/Form(Form const & form) qui fonctionne comme spécifié ar le sujet. Oui Non Exx03 : Au moins, c'est mieux que de faire du café comme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le as contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.	Les notes des formulaires suivent les mêmes contraintes que celles du Bureaucrate (exceptions, 1 = le plus élevé, 150 = le plus bas, et ainsi de suite). Il existe des accesseurs pour les attributs et un opérateur << pour la surcharge			
existe une fonction membre Form:beSigned() qui fonctionne comme décrit par le sujet. Aviste une fonction Bureaucrat:signForm() qui fonctionne comme décrit par le sujet. Oui Non Non Non Non Non Non Non Non				
existe une fonction membre Form:beSigned() qui fonctionne comme décrit par le sujet. Oui Non SEXO2: Non, vous avez besoin du formulaire 28B, pas du 28C Oui Non Exo2: Non, vous avez besoin du formulaire 28B, pas du 28C Oui Non Exo2: Non, vous avez besoin du formulaire 28B, pas du 28C Oui Non Exo3: Non, vous avez besoin du formulaire 28B, pas du 28C Oui Non Exo4: In faction principale doit content suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le les contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice. Se formulaires qui font réellement quelque chose existe un Makeffle qui se compile en utilisant les indicateurs appropriés. Existe des formes concrètes qui sont conformes aux spécifications de la matièrre (notes quises, noms et actions). In se prennent qu'un seul paramètre dans leur constructeur, qui est leur cible. existe une méthode Porm:execute(Bursaucrat const & executor) qui fonctionne comme destinates de formes executes exercices executes des formes qui exécute uniquement faction. se deux techniques sont valables. existe un Bursaucrat::executeForm(Form const & form) qui fonctionne comme spécifié ar le sujet. Oui Non Exx03: Au moins, c'est mieux que de faire du café comme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le les contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.				
Oui Non Ex02: Non, vous avez besoin du formulaire 28B, pas du 28C Oui Non Ex02: Non, vous avez besoin du formulaire 28B, pas du 28C Domme d'habitude, le fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le ses contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice. Se formulaires qui font réellement quelque chose existe un Makeffle qui se compile en utilisant les indicateurs appropriés. Existe des formes conorêtes qui sont conformes aux spécifications de la matière (notes quises, noms et actions). In experment qu'un seul paramètre dans leur constructeur, qui est leur cible. Existe une méthode est pure et les vérifications de qualité sont implémentées dans chaque bus-classe, soit cette méthode effectue les vérifications, puis appeile une autre méthode aux les dévivée qui exécute uniquement l'action. Es deux techniques sont valiables. Existe un Bureaucrat:-executeForm(Form const & form) qui fonctionne comme spécifié ar le sujet. Oui Non Ex03: Au moins, c'est mieux que de faire du café comme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le sas contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.	ostream qui affiche l'état complet du formulaire.			
Ex02 : Non, vous avez besoin du formulaire 28B, pas du 28C omme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le sa contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice. es formulaires qui font réellement quelque chose existe un Makefile qui se compile en utilisant les indicateurs appropriés. existe un Makefile qui se compile en utilisant les indicateurs appropriés. existe des formes concrètes qui sont conformes aux spécifications de la matière (notes quisses, noms et actions). In prennent qu'un seul paramètre dans leur constructeur, qui est leur cible, existe une méthode Form: execute (Bureaucrat const & executor) qui fonctionne comme prédité par le sujet. Dui cette méthode est pure et les vérifications de qualité sont implémentées dans chaque pus-classe, soit cette méthode effectue les vérifications, puis appelle une autre méthode ans la classe dérivée qui exécute uniquement l'action. es deux techniques sont valables. existe un Bureaucrat: executeForm(Form const & form) qui fonctionne comme spécifié ar le sujet. Oui Non Ex03 : Au moins, c'est mieux que de faire du café comme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le as contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.	ll existe une fonction membre Form::beSigned() qui fonctionne comme décrit par le sujet. Il existe une fonction Bureaucrat::signForm() qui fonctionne comme décrit par le sujet.			
Ex02 : Non, vous avez besoin du formulaire 28B, pas du 28C omme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le as contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice. es formulaires qui font réellement quelque chose existe un Makefflie qui se compile en utilisant les indicateurs appropriés. existe des formes concrètes qui sont conformes aux spécifications de la matière (notes quises, noms et actions). si ne prement qu'un seul paramètre dans leur constructeur, qui est leur cible, existe une méthode Form: execute(Bureaucrat const & executor) qui fonctionne comme pécifié par le sujet. di cette méthode est pure et les vérifications de qualité sont implémentées dans chaque bous-classe, soit cette méthode effectue les vérifications, puis appelle une autre méthode ans la classe dérivée qui exécute uniquement l'action. es deux techniques sont valables. existe un Bureaucrat::executeForm(Form const & form) qui fonctionne comme spécifié ar le sujet. Oui Non EXO3 : Au moins, c'est mieux que de faire du café comme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le as contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.				
omme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le as contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice. Se contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice. Se contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice. Se contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.	Oui	Non		
existe un Makefile qui se compile en utilisant les indicateurs appropriés. existe des formes concrètes qui sont conformes aux spécifications de la matière (notes requises, noms et actions). In perennent qu'un seul paramètre dans leur constructeur, qui est leur cible. Existe une méthode Form::execute(Bureaucrat const & executor) qui fonctionne comme desiste une et les vérifications de qualité sont implémentées dans chaque dous-classe, soit cette méthode effectue les vérifications, puis appelle une autre méthode ans la classe dérivée qui exécute uniquement l'action. Existe un Bureaucrat::executeForm(Form const & form) qui fonctionne comme spécifié ar le sujet. Oui Non Ex03: Au moins, c'est mieux que de faire du café comme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le as contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.	Comme d'habitude, la fonction principale doit contenir suffisamment de tests pour p	orouver que le programme fonctionne comme prévu. Dans le		
existe des formes concrètes qui sont conformes aux spécifications de la matière (notes rquises, noms et actions). In existe une méthode Form::execute(Bureaucrat const & executor) qui fonctionne comme pécifié par le sujet. Décifié par le sujet. Doui Non Ex03: Au moins, c'est mieux que de faire du café Douis des pour prouver que le programme fonctionne comme prévu. Dans le as contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.	Des formulaires qui font réellement quelque chose			
existe une méthode Form::execute(Bureaucrat const & executor) qui fonctionne comme bécifié par le sujet. oit cette méthode est pure et les vérifications de qualité sont implémentées dans chaque pus-classe, soit cette méthode effectue les vérifications, puis appelle une autre méthode ans la classe dérivée qui exécute uniquement l'action. es deux techniques sont valables. existe un Bureaucrat::executeForm(Form const & form) qui fonctionne comme spécifié ar le sujet. Oui Non Ex03: Au moins, c'est mieux que de faire du café omme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le as contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.	l existe un Makefile qui se compile en utilisant les indicateurs appropriés.			
s ne prennent qu'un seul paramètre dans leur constructeur, qui est leur cible. existe une méthode Form::execute(Bureaucrat const & executor) qui fonctionne comme pécifié par le sujet. oit cette méthode est pure et les vérifications de qualité sont implémentées dans chaque pous-classe, soit cette méthode effectue les vérifications, puis appelle une autre méthode ans la classe dérivée qui exécute uniquement l'action. es deux techniques sont valables. existe un Bureaucrat::executeForm(Form const & form) qui fonctionne comme spécifié ar le sujet. Oui Non EXO3: Au moins, c'est mieux que de faire du café omme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le as contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.	existe des formes concrètes qui sont conformes aux spécifications de la matière (notes		
existe une méthode Form::execute(Bureaucrat const & executor) qui fonctionne comme pécifié par le sujet. oit cette méthode est pure et les vérifications de qualité sont implémentées dans chaque pus-classe, soit cette méthode effectue les vérifications, puis appelle une autre méthode ans la classe dérivée qui exécute uniquement l'action. es deux techniques sont valables. existe un Bureaucrat::executeForm(Form const & form) qui fonctionne comme spécifié ar le sujet. Oui Non Ex03 : Au moins, c'est mieux que de faire du café omme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le as contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.	equises, noms et actions).			
pécifié par le sujet. oit cette méthode est pure et les vérifications de qualité sont implémentées dans chaque pous-classe, soit cette méthode effectue les vérifications, puis appelle une autre méthode ans la classe dérivée qui exécute uniquement l'action. es deux techniques sont valables. existe un Bureaucrat::executeForm(Form const & form) qui fonctionne comme spécifié ar le sujet. Oui Non Ex03 : Au moins, c'est mieux que de faire du café omme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le as contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.	s ne prennent qu'un seul paramètre dans leur constructeur, qui est leur cible.			
oit cette méthode est pure et les vérifications de qualité sont implémentées dans chaque ous-classe, soit cette méthode effectue les vérifications, puis appelle une autre méthode ans la classe dérivée qui exécute uniquement l'action. es deux techniques sont valables. existe un Bureaucrat::executeForm(Form const & form) qui fonctionne comme spécifié ar le sujet. Oui Non Ex03 : Au moins, c'est mieux que de faire du café comme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le as contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.	existe une méthode Form::execute(Bureaucrat const & executor) qui fonctionne c	omme		
cous-classe, soit cette méthode effectue les vérifications, puis appelle une autre méthode ans la classe dérivée qui exécute uniquement l'action. es deux techniques sont valables. existe un Bureaucrat::executeForm(Form const & form) qui fonctionne comme spécifié ar le sujet. Oui Non Ex03: Au moins, c'est mieux que de faire du café comme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le as contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.	pécifié par le sujet.			
es deux techniques sont valables. existe un Bureaucrat::executeForm(Form const & form) qui fonctionne comme spécifié ar le sujet. Oui Non Ex03 : Au moins, c'est mieux que de faire du café comme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le as contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.	Soit cette méthode est pure et les vérifications de qualité sont implémentées dans	chaque		
existe un Bureaucrat::executeForm(Form const & form) qui fonctionne comme spécifié ar le sujet. Oui Non Ex03 : Au moins, c'est mieux que de faire du café omme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le as contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.	ous-classe, soit cette méthode effectue les vérifications, puis appelle une autre me	éthode		
existe un Bureaucrat::executeForm(Form const & form) qui fonctionne comme spécifié ar le sujet. Oui Non Ex03 : Au moins, c'est mieux que de faire du café omme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le as contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.	lans la classe dérivée qui exécute uniquement l'action.			
Oui Non Ex03 : Au moins, c'est mieux que de faire du café omme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le as contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.	Ces deux techniques sont valables.			
Oui Non Ex03 : Au moins, c'est mieux que de faire du café omme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le as contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.	existe un Bureaucrat::executeForm(Form const & form) qui fonctionne comme spe	écifié		
Ex03 : Au moins, c'est mieux que de faire du café omme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le as contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.	ar le sujet.			
omme d'habitude, la fonction principale doit contenir suffisamment de tests pour prouver que le programme fonctionne comme prévu. Dans le as contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.	Oui	Non		
as contraire, ne notez pas cet exercice. Si une classe non-interface n'est pas conforme à la forme canonique orthodoxe, ne notez pas cet exercice.	·			
terne	and the state of t	2 2 Common Com		
teme				
	nterne			

	tion makeForm() qui fonctionne comme spécific	ė par le sujet.	
	Oui	Non	
Bon envoi			
La fonction makeFor	m() doit utiliser une sorte de tableau de pointeun de formulaires.	rs vers des fonctions membres	
S'il utilise une métho	de impure, comme les branches if/elseif/elseif/	else, ou d'autres	
trucs laids comme ça	a, veuillez considérer cela comme faux.		
	Oui	Non	
Travail vide	her le drapeau correspondant à la défense D'accord Aucun fichier auteur W Compilation invalide	Projet exceptionnel Norme Cheat d Crash	
	Fuites	I Fonction interdite	
Conclus Laissez un comment Проверкаоф МСК. Пиррас о	aire sur cette évaluation		

Politique de confidentialité (https://signin.intra.42.fr/legal/terms/5)

Conditions d'utilisation de la vidéosurveillance (https://signin.intra.42.fr/legal/terms/1)

Règlement intérieur (https://signin.intra.42.fr/legal/terms/4)

Déclaration relative à l'utilisation des cookies (https://signin.intra.42.fr/legal/terms/2)

Conditions générales d'utilisation du site (https://signin.intra.42.fr/legal/terms/6)

Mentions légales (https://signin.intra.42.fr/legal/terms/3)