PLAN DE TEST :

Projet : Cacalzone

Groupe : Anis LAFRAD / Gabriel GASNIER / Safiatou DIALLO

TestPizza.java

Méthode de test	Actions	Attendu
testConstructeurEtGetters	Instancier une Pizza "Reine" de type VIANDE.	Le nom est "Reine", le type est VIANDE, les listes d'ingrédients et d'évaluations sont vides, le prix est 0.0 (car pas d'ingrédients).
testSetNomValide	Modifier le nom de la pizza par "Royale".	getNom() renvoie "Royale".
testSetNomInvalide	Essayer de modifier le nom avec null ou une chaîne vide "".	Une IllegalArgumentException est levée.
testSetTypePizza	Modifier le type de la pizza pour VEGETARIENNE.	getTypePizza() renvoie VEGETARIENNE.
testAjouterIngredientValide	Ajouter un ingrédient "Tomate" (prix 1.0) compatible avec le type.	La méthode renvoie true. La liste des ingrédients contient "Tomate".
testAjouterIngredientDoublon	Ajouter deux fois le même ingrédient "Tomate".	Le deuxième appel renvoie false. La liste ne contient qu'une seule fois l'ingrédient.
testAjouterIngredientInterdit	Essayer d'ajouter un ingrédient "Jambon" configuré comme interdit pour le type VIANDE (test théorique) ou VEGETARIENNE.	La méthode renvoie false. L'ingrédient n'est pas ajouté à la liste.
testCalculPrixMinimal	Ajouter des ingrédients pour un total de 7.30€. Vérifier le calcul (Marge 40% + Arrondi au décime sup).	Le prix minimal calculé doit être 10.30€ (7.30 * 1.4 = 10.22 -> arrondi 10.30).
testSetPrixValide	Définir un prix de vente manuel supérieur au prix minimal calculé.	La méthode renvoie true. getPrix() renvoie le prix manuel défini.
testSetPrixInvalide	Définir un prix de vente manuel inférieur au prix minimal calculé.	La méthode renvoie false. getPrix() continue de renvoyer le prix minimal (ou l'ancien prix valide).
testPhoto	Modifier et récupérer le chemin de la photo.	getPhoto() renvoie le chemin défini via setPhoto().

TestIngrédient.java

Méthode de test	Actions	Attendu
testConstructeur	Instancier un ingrédient "Olive" avec un prix de 0.20€.	Le nom est "Olive", le prix est 0.20, et la liste des types interdits est initialisée mais vide.
testSetNom	Modifier le nom de l'ingrédient pour "Olive Noire".	getNom() renvoie "Olive Noire".
testSetPrix	Modifier le prix de l'ingrédient à 0.30€.	getPrix() renvoie 0.30.
testAddTypePizzaInterdit	Ajouter une interdiction pour le type VIANDE.	La liste des interdits (getTypesPizzaInterdits) contient VIANDE et sa taille est de 1.
testAddTypePizzaInterditDoublon	Ajouter deux fois l'interdiction pour le type VIANDE.	La liste ne contient qu'une seule fois VIANDE (propriété des Sets). La taille reste à 1.
testRemoveTypePizzaInterdit	Ajouter l'interdiction VEGETARIENNE puis appeler removeTypePizzaInterdit pour ce même type.	L'interdiction est retirée, la liste redevient vide.
testSetTypesPizzaInterdits	Créer un nouveau Set contenant VIANDE et REGIONALE, et l'affecter via le setter.	getTypesPizzaInterdits() renvoie le nouveau Set contenant exactement ces deux éléments.

TestUtilisateur.java

Méthode de test	Actions	Attendu
testConstructeurEtGetters	Instancier une classe concrète étendant Utilisateur avec un email "test@mail.com", un mot de passe "1234" et des infos personnelles valides.	L'instance est créée avec succès. getEmail() renvoie "test@mail.com", getMotDePasse() renvoie "1234", et getInfos() renvoie l'objet infos passé en paramètre.
testEstPizzaioloVrai	Instancier une implémentation concrète où estPizzaiolo renvoie true.	La méthode estPizzaiolo() renvoie true.
testEstPizzaioloFaux	Instancier une implémentation concrète où estPizzaiolo renvoie false.	La méthode estPizzaiolo() renvoie false.

TestCompteClient.java

Méthode de test	Actions	Attendu
testConstructeurEtInitialisation	Instancier un CompteClient avec un email "client@mail.fr", un mot de passe "mdp" et des infos valides.	L'instance est créée. Les attributs hérités (email, mot de passe) sont corrects. La liste historiqueCommandes est instanciée (non null) mais vide (taille 0).
testEstPizzaiolo	Appeler la méthode estPizzaiolo().	La méthode renvoie false.

testAjouterCommande	Créer une Commande associée à ce client, puis l'ajouter via ajouterCommande().	La taille de la liste getHistoriqueCommandes() passe à 1. La liste contient bien l'objet commande ajouté.

TestComptePizzaiolo.java

Méthode de test	Actions	Attendu
testConstructeur	Instancier un ComptePizzaiolo avec l'email "chef@pizza.fr", le mot de passe "admin" et des infos valides.	L'instance est créée sans erreur. Les getters (getEmail, getMotDePasse) renvoient les valeurs passées en paramètre.
testEstPizzaiolo	Appeler la méthode estPizzaiolo() sur l'instance.	La méthode renvoie true (contrairement à un compte client).

TestInformationPersonnelle.java

Méthode de test	Actions	Attendu
testConstructeurBasique	Instancier avec seulement nom ("Doe") et prénom ("John").	Le nom est "Doe", prénom "John". L'adresse est initialisée à une chaîne vide "" (et non null). L'âge est initialisé à 0.
testConstructeurComplet	Instancier avec tous les paramètres ("Doe", "John", "Brest", 25).	Tous les champs (nom, prénom, adresse, âge) correspondent exactement aux valeurs fournies.
testSetAgeValide	Appeler setAge(30) sur une personne.	getAge() renvoie 30.
testSetAgeInvalide	Appeler setAge(-5) ou setAge(0) sur une personne ayant déjà un âge valide (ex: 25).	La modification est ignorée. getAge() renvoie toujours l'ancienne valeur (25).
testSetAdresseValide	Appeler setAdresse("Paris").	getAdresse() renvoie "Paris".
testSetAdresseInvalide	Appeler setAdresse(null) sur une personne ayant déjà une adresse (“Brest”)	La modification est ignorée. getAdresse() renvoie toujours l'ancienne valeur ("Brest").
testEquals	Créer deux instances avec exactement les mêmes nom, prénom, adresse et âge.	equals() renvoie true.
testNotEquals	Créer deux instances différant par au moins un attribut (ex: âge différent).	equals() renvoie false.

TestCommande.java

Méthode de test	Actions	Attendu
testConstructeur	Instancier une commande avec un CompteClient valide.	La commande est créée. L'état est CREEE. La liste des pizzas est vide. Le client est bien associé. La date n'est pas nulle.
testAjouterPizzaValide	Ajouter 1 pizza "Reine". Puis ajouter 2 autres "Reine" (même objet).	La méthode renvoie true. La map contient la pizza "Reine" avec une quantité de 3.
testAjouterPizzaInvalide	Essayer d'ajouter une pizza null ou avec une quantité de 0 ou -1.	La méthode renvoie false. La pizza n'est pas ajoutée.
testAjouterPizzaException	Valider une commande (état VALIDEE), puis essayer d'ajouter une pizza.	Une CommandeException est levée car on ne peut plus modifier une commande validée.
testValiderSucces	Ajouter une pizza à une commande CREEE puis appeler valider().	La méthode renvoie true. L'état passe à VALIDEE.
testValiderEchec	Appeler valider() sur une commande vide (sans pizzas).	La méthode renvoie false. L'état reste à CREEE.
testTraiter	Valider une commande, puis appeler traiter().	L'état passe de VALIDEE à TRAITEE.
testAnnulerDepuisCreee	Ajouter une pizza à une commande CREEE, puis appeler annuler().	L'état passe à ANNULEE. La liste des pizzas est vidée.
testAnnulerDepuisValidee	Valider une commande, puis appeler annuler().	L'état passe à ANNULEE. La liste des pizzas est vidée.
testGetPrixTotal	Ajouter une pizza A (12€) qté 1 et une pizza B (10€) qté 2.	getPrixTotal() renvoie 32.0€ (12 + 10*2).

TestMenu.java

Méthode de test	Actions	Attendu
testConstructeur	Instancier un Menu.	Les listes (pizzas, ingrédients, commandes) sont instanciées mais vides. La liste des utilisateurs contient déjà 1 élément (le compte par défaut du pizzaïolo "Mario").
testAuthentifierSucces	Appeler authentifier("chef@pizza.fr", "admin").	La méthode renvoie l'objet Utilisateur correspondant (ne renvoie pas null).
testAuthentifierEchecMdp	Appeler authentifier("chef@pizza.fr", "mauvaisMdp").	La méthode renvoie null.
testAuthentifierEchecEmail	Appeler authentifier("inconnu@pizza.fr", "admin").	La méthode renvoie null.
testTrouverUtilisateurExiste	Appeler trouverUtilisateur("chef@pizza.fr").	La méthode renvoie l'utilisateur correspondant.
testTrouverUtilisateurInconnu	Appeler	La méthode renvoie null.

	trouverUtilisateur("personne@test.fr").	
testAjouterUtilisateur	Créer un nouveau CompteClient et l'ajouter via ajouterUtilisateur.	La taille de la liste des utilisateurs passe à 2. L'authentification avec ce nouveau compte fonctionne.
testAjouterCommande	Créer une commande et l'ajouter via ajouterCommande.	La liste getCommandesGlobales() n'est plus vide et contient l'objet ajouté.
testGettersCollections	Vérifier les getters getPizzas et getIngredients.	Ils renvoient des Set non nuls (vides au départ).

TestEvaluation.java

Méthode de test	Actions	Attendu
testConstructeurValide	Instancier une évaluation avec une note de 4, un commentaire "Bon" et un email "client@mail.com".	L'instance est créée. getNote() renvoie 4, getCommentaire() renvoie "Bon", getEmailClient() renvoie l'email.
testConstructeurNoteTropBasse	Instancier avec une note de -2.	La note est automatiquement ramenée à 0.
testConstructeurNoteTropHaute	Instancier avec une note de 10.	La note est automatiquement ramenée à 5.
testConstructeurSansCommentaire	Instancier avec un commentaire null.	getCommentaire() renvoie null. L'objet est valide.
testToStringAvecCommentaire	Appeler toString() sur une évaluation avec commentaire.	La chaîne contient la note et le texte du commentaire (format "Note: X/5 - Commentaire").
testToStringSansCommentaire	Appeler toString() sur une évaluation sans commentaire (null).	La chaîne contient uniquement la note (format "Note: X/5").

TestServiceClient.java

Méthode de test	Actions	Attendu
testInscriptionSucces	Inscrire un nouvel utilisateur avec des données valides.	Renvoie 0. L'utilisateur est ajouté au menu.
testInscriptionDoublon	Inscrire un utilisateur avec un email déjà existant.	Renvoie -1. L'utilisateur n'est pas ajouté en double.
testConnexionSucces	Se connecter avec le compte créé précédemment.	Renvoie true. Le client connecté est bien mémorisé.
testConnexionEchec	Se connecter avec un mauvais mot de passe.	Renvoie false.
testDebuterCommandeNonConnecte	Appeler debuterCommande() sans être connecté.	Lève une NonConnecteException.
testDebuterCommandeConnecte	Se connecter puis appeler debuterCommande().	Renvoie une nouvelle Commande non nulle.
testAjouterPizzaCommande	Débuter une commande, puis ajouter	La commande en cours contient bien la

	une pizza via ajouterPizza().	pizza ajoutée.
testValiderCommande	Débuter une commande, ajouter une pizza, puis appeler validerCommande().	L'état de la commande passe à VALIDEE. Elle est ajoutée à l'historique du client et du menu.
testFiltrePrix	Ajouter un filtre de prix max (12€). Appeler selectionPizzaFiltres().	La liste retournée ne contient que des pizzas dont le prix est <= 12€.
testFiltreIngredient	Ajouter un filtre d'exclusion "Jambon".	La liste retournée ne contient aucune pizza ayant "Jambon" comme ingrédient.

TestServicePizzaiolo.java

Méthode de test	Actions	Attendu
testCreerPizzaSucces	Créer une pizza "Royale" de type VIANDE.	La méthode renvoie une instance non nulle. La pizza est ajoutée au menu.
testCreerPizzaDoublon	Créer une pizza "Royale", puis essayer d'en créer une deuxième avec le même nom (peu importe la casse).	La deuxième tentative renvoie null. Le menu ne contient qu'une seule pizza "Royale".
testCreerIngredientSucces	Créer un ingrédient "Olive" à 0.5€.	La méthode renvoie 0. L'ingrédient est ajouté au menu.
testCreerIngredientInvalide	Essayer de créer un ingrédient avec un nom vide ou un prix négatif.	La méthode renvoie un code d'erreur (-1 ou -3). L'ingrédient n'est pas ajouté.
testInterdireIngredient	Créer un ingrédient "Jambon". L'interdire pour le type VEGETARIENNE.	La méthode renvoie true. L'ingrédient possède bien cette interdiction.
testAjouterIngredientPizza	Ajouter un ingrédient autorisé à une pizza.	La méthode renvoie 0. La pizza contient l'ingrédient.
testAjouterIngredientInterdit	Essayer d'ajouter un ingrédient interdit (voir test précédent) à une pizza du type concerné.	La méthode renvoie -3 (erreur). L'ingrédient n'est pas ajouté.
testCommandeNonTraitees	Avoir une commande VALIDEE dans le menu. Appeler commandeNonTraitees().	La méthode renvoie une liste contenant cette commande. L'état de la commande passe à TRAITEE.
testBeneficeToutesCommandes	Avoir deux commandes traitées générant chacune 5€ de bénéfice.	beneficeToutesCommandes() renvoie 10.0.

TestServiceSauvegarde.java

Méthode de test	Actions	Attendu
testSauvegarderEtCharger	Ajouter un ingrédient "Tomate" au menu. Sauvegarder dans "test.ser". Vider manuellement la liste des ingrédients du menu. Charger depuis "test.ser".	Après le chargement, la liste des ingrédients contient à nouveau "Tomate". Le mécanisme de persistance fonctionne.
testChargerFichierInexistant	Tenter de charger un fichier "bidon.ser" qui n'existe pas sur le disque.	Une IOException est levée.

testPersistanceReferenceMenu	Sauvegarder un menu rempli. Charger les données. Comparer l'objet menu avant et après.	L'objet Menu reste la même instance en mémoire (référence identique), mais son contenu interne (les listes) a été mis à jour.

TestCommandeException.java

Méthode de test	Actions	Attendu
testConstructeurDefaut	Instancier l'exception avec le constructeur vide new CommandeException().	L'instance est créée. La méthode getMessage() renvoie null.
testConstructeurAvecMessage	Instancier l'exception avec un message précis : new CommandeException("Erreur critique").	L'instance est créée. La méthode getMessage() renvoie exactement la chaîne "Erreur critique".