Unit Tests

Unit tests for database

Precondition: An in-memory database is created

Postcondition: The in-memory database created in the Pre-condition should is closed and deleted

ID	Test Case Description	Precondition	Input	Expected Output	Actual Output	Test Result	Test Comments
1	AddCardEntity: Insert a new card entity to the database. THe card should be inserted into the databse when the correct Dao method is called	There is a CardEntity object	cardEntityDao.insert(cardEntity);	The card entity is inserted	The card entity is inserted	PASS	
2	SearchCardByName: Search cards that has the given name. It should return a list of cards with the given name, or just an empty list if there is no such cards. The search should be case insensitive	There is a CardEntity object with the name “sam“ in the database	val cards = cardEntityDao. getCardsByName("Sam");	cards contains and only contains card entity objects with name “sam “ (case insensitive)	cards contains and only contains one card entity objects with name “sam“	PASS	
3	GetAllCardsOrderByName: Order cards entity object by name. It returns a list of card entity order by name in the ascending order (case insensitive)	There are 5 cards enetities in the database, with the names “sam“, ”jack”, “peter “, “adam“, “Zerg“	val cards = cardEntityDao. getAllCardsOrderByName();	return a list of cards entities in the order of “adam“, “jack“, “peter”, “sam“, “Zerg“	return a list of cards entities in the order of “adam“, “jack“, “peter”, “sam“, “Zerg“	PASS	
4	InsertCardListCrossReference: Insert a cross reference entity for the association between card entity and list entity	There is a cardListCrossRef object with cardId and listId	cardListCrossRefDao.insert (cardListCrossRef);	The cardListCrossRef object is inserted into the database	The cardListCrossRef object is inserted into the database	PASS	
5	GetListsWithCards: Get all lists with cards in each list	There are two list entities object, named “first list“ and “second list“ in the database. The relationship between list entity and cards entity is defiend in CardListCrossReference table. ”First list” has two card entities with cardId=50 and cardId=100. The “Second list“ has no cards belong to it	val results: List<ListWithCardsEntity> = listEntityDao.getListWithCards();	Return “First list“ with two cards enetities with cardId=50 and cardId=100. And “Second list“ with no card entitiy	Return “First list“ with two cards enetities with cardId=50 and cardId=100. And “Second list“ with no card entitiy	PASS	
6	DeleteCardById: Delete a card entity with specified card ID	There is a single card entity with ID 10 in the database	cardEntityDao.deleteCardById(10);	The card entity with ID 10 no longer exists in the database	The card entity with ID 10 no longer exists in the database	PASS	
7	DeleteCardWithCrossRefByCardId: When a card entity is deleted from the database, all the associated cross reference of that entity should also be deleted	In the database, there is: • a card entity with ID 10 • a cardListCrossRef entity with card entity ID 10 and list ID 100 • a cardTagCrossRef entity with card entity ID 10 and tag ID 200	appRepository. deleteCardAndCrossRefByCardId (10);	the card entity with ID 10, the cardListCrossRef entity with card entity ID 10 and list ID 100, and the cardTagCrossRef entity with card entity ID 10 and tag ID 200 are all deleted	the card entity with ID 10, the cardListCrossRef entity with card entity ID 10 and list ID 100, and the cardTagCrossRef entity with card entity ID 10 and tag ID 200 are all deleted	PASS	
8	GetListWithCardsByListId: Given a list ID, get the list with all cards entities belong to the list	Threre is a list with ID 100 in the database. There are 5 card entities belong to the list	val listWithCardsEntity: ListWithCardsEntity = listEntityDao. getListWithCardsByListId(100);	Return the list whose ID is 100 with all entities belong to it	Return the list whose ID is 100 with all entities belong to it	PASS	

9	GetCardByTagIds: given a variable number of tag IDs, get all cards that are assoiciated with the tag IDs in OR relationship	In the database, there are three card entities with ID 1, 2, 3, respectively. There are also 2 tags with IDs 100 and 200. Card Entity with ID 1 has tags with ID 100 and 200. Card Entity with ID 2 has tags with ID 100. Card Entity with ID 3 has tags with 200	val result = appRepository. getCardByTagIds(100,200);	Returns all three cards entities	Returns all three cards entities	**PASS**	