Class: InventorySystem						
Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
run	1	The main application loop starts and displays the main menu.	The run() method is called.	The main menu is displayed to the console.	The main menu is displayed to the console.	P
Class: Card						
Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
getName	1	Returns the correct name of the Card.	Card created with name "Pikachu"	Pikachu	Pikachu	P
getBaseValue	1	Returns the correct base value of the Card.	Card created with base value 25.50	25.5	25.5	P
getRarity	1	Returns the correct Rarity enum constant.	Card created with Rarity.RARE	Rarity.RARE	Rarity.RARE	P
getVariant	1	Returns the correct Variant enum constant.	Card created with Variant.FULL_ART	Variant.FULL_ART	Variant.FULL_ART	P
	1	Returns correct value for a normal variant.	Base value: 50.0, Variant: Variant.NORMAL	50	50	P
	2	Returns correct value for Extended-art variant.	Base value: 50.0, Variant: Variant.EXTENDED_ART	75	75	P
	3	Returns correct value for Full-art variant.	Base value: 50.0, Variant: Variant.FULL_ART	100	100	P
getCalculatedValue	4	Returns correct value for Alt-art variant.	Base value: 50.0, Variant: Variant.ALT_ART	150	150	P
Enum: Rarity						
Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
	1	Returns correct display name for RARE.	Rarity.RARE.getDisplayName()	Rare	Rare	P
getDisplayName	2	Returns correct display name for COMMON.	Rarity.COMMON.getDisplayName()	Common	Common	P
	1	Returns correct enum for valid input 1.	Rarity.fromInt(1)	Rarity.COMMON	Rarity.COMMON	P
	2	Returns correct enum for valid input 4.	Rarity.fromInt(4)	Rarity.LEGENDARY	Rarity.LEGENDARY	P
	3	Returns null for invalid zero input.	Rarity.fromInt(0)	null	null	P
fromInt	4	Returns null for invalid out-of-bounds input.	Rarity.fromInt(5)	null	null	P
Enum: Variant						
Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
getDisplayName	1	Returns correct display name for FULL_ART.	Variant.FULL_ART.getDisplayName()	Full-art	Full-art	P
	1	Returns correct multiplier for EXTENDED_ART.	Variant.EXTENDED_ART.getMultiplier()	1.5	1.5	P
getMultiplier	2	Returns correct multiplier for NORMAL.	Variant.NORMAL.getMultiplier()	1	1	P
	1	Returns correct enum for valid input 1.	Variant.fromInt(1)	Variant.NORMAL	Variant.NORMAL	P
	2	Returns correct enum for valid input 3.	Variant.fromInt(3)	Variant.FULL_ART	Variant.FULL_ART	P
	3	Returns null for invalid negative input.	Variant.fromInt(-1)	null	null	P
fromInt	4	Returns null for invalid out-of-bounds input.	Variant.fromInt(5)	null	null	P
Class: Binder						
Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
getName	1	Returns the correct name of the Binder.	Binder created with name "Trade Binder"	Trade Binder	Trade Binder	P
getCards	1	Returns a defensive copy of the cards list.	Binder with 2 cards. getCards() is called.	New ArrayList with 2 cards	New ArrayList with 2 cards	P
	1	Returns 0 for a new binder.	New binder created.	0	0	P
getCardCount	2	Returns correct count after adding cards.	5 cards added to binder.	5	5	P
	1	Returns false when binder is not full.	Binder with 19 cards.	FALSE	FALSE	P
isFull	2	Returns true when binder is full.	Binder with 20 cards.	TRUE	TRUE	P
	1	Adds a card to a non-full binder.	Add card to binder with 10 cards.	TRUE	TRUE	P
addCard	2	Fails to add a card to a full binder.	Add card to binder with 20 cards.	FALSE	FALSE	P
	1	Fails to remove card at negative index.	removeCard(-1)	null	null	P
	2	Removes an existing card successfully.	removeCard(0) on a binder with 1 card.	The removed Card object	The removed Card object	P
removeCard	3	Fails to remove card at out-of-bounds index.	removeCard(5) on a binder with 5 cards.	null	null	P
Class: Deck						
Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
getName	1	Returns the correct name of the Deck.	Deck created with name "My Deck"	My Deck	My Deck	P
getCards	1	Returns a defensive copy of the cards list.	Deck with 3 cards. getCards() is called.	New ArrayList with 3 cards	New ArrayList with 3 cards	P
getCardCount	1	Returns correct count after adding cards.	7 cards added to deck.	7	7	P
	1	Returns false when deck is not full.	Deck with 9 cards.	FALSE	FALSE	P
isFull	2	Returns true when deck is full.	Deck with 10 cards.	TRUE	TRUE	P
	1	Returns true if card with same name exists.	Deck has "Pikachu". containsCard("Pikachu")	TRUE	TRUE	P
	2	Returns true for case-insensitive check.	Deck has "Pikachu". containsCard("pikachu")	TRUE	TRUE	P
containsCard	3	Returns false if card does not exist.	Deck has "Pikachu". containsCard("Charmander")	FALSE	FALSE	P
	1	Adds a unique card to a non-full deck.	Add "Charmander" to deck with 5 cards.	TRUE	TRUE	P
	2	Fails to add a card if deck is full.	Add "Squirtle" to deck with 10 cards.	FALSE	FALSE	P
addCard	3	Fails to add a card with a duplicate name.	Add "Pikachu" to deck that already has a "Pikachu".	FALSE	FALSE	P
	1	Removes an existing card successfully.	removeCard(2) on a deck with 3 cards.	The removed Card object	The removed Card object	P
removeCard	2	Fails to remove card at out-of-bounds index.	removeCard(10) on a deck with 10 cards.	null	null	P
Class: CollectionManager						
Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
	1	Finds an existing card by name.	findCard("Charizard") when it exists.	Card object for "Charizard"	Card object for "Charizard"	P
findCard	2	Returns null for a non-existent card.	findCard("Mewtwo") when it does not exist.	null	null	P
	1	Adds a new, unique card type successfully.	Add "Bulbasaur" for the first time.	TRUE	TRUE	P
addNewCard	2	Fails to add a card with a duplicate name.	Add "Bulbasaur" again.	FALSE	FALSE	P
	1	Increases the count of an existing card.	Card "A" count is 1. increaseCount("A", 5)	true (count becomes 6)	true (count becomes 6)	P
increaseCount	2	Fails to increase count of a non-existent card.	increaseCount("B", 5)	FALSE	FALSE	P
	1	Decreases count when sufficient copies exist.	Card "C" count is 10. decreaseCount("C", 3)	true (count becomes 7)	true (count becomes 7)	P
decreaseCount	2	Fails to decrease count below zero.	Card "D" count is 2. decreaseCount("D", 5)	false (count remains 2)	false (count remains 2)	P
	1	Returns true if card count > 0.	Card "E" has count 1.	TRUE	TRUE	P
isCardAvailable	2	Returns false if card count is 0.	Card "F" has count 0.	FALSE	FALSE	P
getCardTypes	1	Returns a list of all unique card objects.	Collection has 3 unique cards.	ArrayList of size 3	ArrayList of size 3	P
	1	Returns the correct count of a card.	Card "G" has count 8. getCardCount("G")	8	8	P
getCardCount	2	Returns 0 for a non-existent card.	getCardCount("H")	0	0	P
Class: BinderManager						
Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
findBinder	1	Finds an existing binder.	findBinder("My Trades") when it exists.	Binder object	Binder object	P
	2	Returns null for non-existent binder.	findBinder("Lost Binder") when it does not exist.	null	null	P
createBinder	1	Creates a new, unique binder.	createBinder("New Binder")	TRUE	TRUE	P
	2	Fails to create a binder with a duplicate name.	createBinder("New Binder") again.	FALSE	FALSE	P
	1	Deletes an existing binder.	deleteBinder("Old Binder")	TRUE	TRUE	P
deleteBinder	2	Fails to delete a non-existent binder.	deleteBinder("Non-existent Binder")	FALSE	FALSE	P
	1	Adds available card to valid binder.	addCardToBinder("Pikachu", "Trades")	0 (Success)	0 (Success)	P
	2	Fails if card or binder does not exist.	addCardToBinder("Mew", "Game Binder")	1 (Not Found)	1 (Not Found)	P

addCardToBinder	3	Fails if no copies of card are available.	Card "Zapdos" has count 0. addCardToBinder(...)	2 (No Copies)	2 (No Copies)	P
	1	Removes a card from a valid index.	Remove card at index 0 from non-empty binder.	TRUE	TRUE	P
removeCardFromBinder	2	Fails to remove from an invalid index.	Remove card at index 99 from binder.	FALSE	FALSE	P
performTrade	1	Successfully completes a valid trade.	Trade card at index 0 for a new valid card.	TRUE	TRUE	P
getBinders	1	Returns a list of all binder objects.	Manager has 4 binders.	ArrayList of size 4	ArrayList of size 4	P
Class: DeckManager						
Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
	1	Finds an existing deck.	findDeck("Main Deck") when it exists.	Deck object	Deck object	P
findDeck	2	Returns null for non-existent deck.	findDeck("Side Deck") when it does not exist.	null	null	P
	1	Creates a new, unique deck.	createDeck("New Deck")	TRUE	TRUE	P
createDeck	2	Fails to create a deck with a duplicate name.	createDeck("New Deck") again.	FALSE	FALSE	P
	1	Deletes an existing deck.	deleteDeck("Old Deck")	TRUE	TRUE	P
deleteDeck	2	Fails to delete a non-existent deck.	deleteDeck("Non-existent Deck")	FALSE	FALSE	P
	1	Adds available card to valid deck.	addCardToDeck("Pikachu", "Main Deck")	0 (Success)	0 (Success)	P
	2	Fails if card is duplicate in deck.	Deck has "Pikachu". addCardToDeck("Pikachu", ...)	4 (Duplicate)	4 (Duplicate)	P
addCardToDeck	3	Fails if deck is full.	Deck has 10 cards. addCardToDeck(...)	3 (Full)	3 (Full)	P
removeCardFromDeck	1	Removes a card from a valid index.	Remove card at index 0 from non-empty deck.	TRUE	TRUE	P
getDecks	1	Returns a list of all deck objects.	Manager has 2 decks.	ArrayList of size 2	ArrayList of size 2	P
Class: Validator						
Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
	1	Returns true for a non-blank string.	isValidName("Test")	TRUE	TRUE	P
	2	Returns false for a blank string.	isValidName(" ")	FALSE	FALSE	P
isValidName	3	Returns false for a null string.	isValidName(null)	FALSE	FALSE	P
	1	Returns true for a positive double.	isPositive(10.5)	TRUE	TRUE	P
	2	Returns false for a negative double.	isPositive(-0.1)	FALSE	FALSE	P
	3	Returns true for a positive integer.	isPositive(1)	TRUE	TRUE	P
isPositive	4	Returns false for a non-positive integer.	isPositive(0)	FALSE	FALSE	P
Class: Inputter						
Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
	1	Returns integer on valid numeric input.	User enters "123"	123	123	P
getIntInput	2	Reprompts on invalid non-numeric input.	User enters "abc", then "45"	45	45	P
getStringInput	1	Returns the raw string entered by the user.	User enters " My Binder "	My Binder	My Binder	P
	1	Returns trimmed string on valid input.	User enters " Test Name "	Test Name	Test Name	P
getValidName	2	Reprompts on blank input.	User enters " ", then "Valid Name"	Valid Name	Valid Name	P
	1	Returns correct enum from valid numeric selection.	User is shown list, enters "3".	Rarity.RARE	Rarity.RARE	P
getValidRarity	2	Reprompts on invalid numeric selection.	User enters "99", then "2".	Rarity.UNCOMMON	Rarity.UNCOMMON	P
getValidVariant	1	Returns correct enum from valid numeric selection.	User enters "4".	Variant.ALT_ART	Variant.ALT_ART	P
	1	Returns double on valid positive input.	User enters "99.99"	99.99	99.99	P
getValidPositiveDouble	2	Reprompts on negative input.	User enters "-10.5", then "5.5"	5.5	5.5	P
	1	Returns int on valid positive input.	User enters "10"	10	10	P
getValidPositiveInt	2	Reprompts on zero input.	User enters "0", then "1"	1	1	P
Class: Display						
Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
getHeader	1	Returns a formatted header string.	getHeader("TEST")	A string with separators and the centered title "TEST"	A string with separators and the centered title "TEST"	P
get...Menu	1	All get...Menu methods return a non-empty string.	Display.getMainMenu()	A valid menu String	A valid menu String	P
getCardDetails	1	Returns a formatted string of card details.	A valid Card object.	A multi-line string with all card attributes.	A multi-line string with all card attributes.	P
getList	1	Returns a formatted, numbered list of items.	An ArrayList of 3 strings.	A numbered list string with a title and 3 items.	A numbered list string with a title and 3 items.	P
Class: Handler						
Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
	1	Displays collection and prompts for details view.	Collection has cards. User selects card #1.	Card #1 details are displayed.	Card #1 details are displayed.	P
	2	Handles user backing out of details view.	Collection has cards. User selects "0" to go back.	No details are displayed; method returns.	No details are displayed; method returns.	P
handleViewCollection	3	Handles an empty collection gracefully.	The collection has no cards.	Prints "Collection is empty."	Prints "Collection is empty."	P
	1	Successfully adds a new, unique card.	User enters valid, unique name "Pikachu", rarity "Rare", variant "Full-art", value "25.0".	Prints "Success! Card 'Pikachu' added." and the card is added to CollectionManager.	Prints "Success! Card 'Pikachu' added." and the card is added to CollectionManager.	P
	2	Fails to add a card with a duplicate name.	User enters "Pikachu" when it already exists in the CollectionManager.	Prints "A card with this name already exists." and no new card is added.	Prints "A card with this name already exists." and no new card is added.	P
handleAddNewCard	3	Reprompts for a blank name and then succeeds.	User enters blank name, then "Squirtle", then valid rarity/variant/value.	Prints "Card name cannot be blank." then "Success! Card 'Squirtle' added."	Prints "Card name cannot be blank." then "Success! Card 'Squirtle' added."	P
	1	Successfully increases the count of a card.	User selects "Pikachu" (count: 2), chooses action (1) Increase, enters amount "3".	Prints "Count updated." and card count in CollectionManager becomes 5.	Prints "Count updated." and card count in CollectionManager becomes 5.	P
	2	Successfully decreases the count of a card.	User selects "Pikachu" (count: 5), chooses action (2) Decrease, enters amount "2".	Prints "Count updated." and card count in CollectionManager becomes 3.	Prints "Count updated." and card count in CollectionManager becomes 3.	P
	3	Fails to decrease count more than available.	User selects "Pikachu" (count: 3), chooses action (2) Decrease, enters amount "5".	Prints "Operation failed..." and card count in CollectionManager remains 3.	Prints "Operation failed..." and card count in CollectionManager remains 3.	P
handleUpdateCardCount	4	Handles user backing out of selection.	User is prompted to select a card and enters "0".	The method returns to the previous menu without changing any state.	The method returns to the previous menu without changing any state.	P
	1	Displays cards in a selected binder.	User selects a binder with 3 cards.	A formatted list of 3 card names is printed.	A formatted list of 3 card names is printed.	P
handleViewBinders	2	Handles selection of an empty binder.	User selects a binder with 0 cards.	Prints "This binder is empty."	Prints "This binder is empty."	P
	1	Successfully creates a new binder with a unique name.	User enters a valid, unique name "Trade Binder".	Prints "Binder created successfully." and the binder is added to BinderManager.	Prints "Binder created successfully." and the binder is added to BinderManager.	P
handleCreateBinder	2	Fails to create a binder with a duplicate name.	User enters "Trade Binder" when it already exists.	The backend prints an error message, and no new binder is added.	The backend prints an error message, and no new binder is added.	P
	1	Successfully deletes an existing binder after confirmation.	User selects "Trade Binder", then confirms with "yes".	Prints "Binder deleted." and the binder is removed from BinderManager.	Prints "Binder deleted." and the binder is removed from BinderManager.	P
handleDeleteBinder	2	Cancels deletion if user does not confirm.	User selects "Trade Binder", then enters "no".	Prints "Deletion canceled." and the binder remains.	Prints "Deletion canceled." and the binder remains.	P
	1	Successfully adds an available card to a binder.	User selects "Trade Binder", then selects "Pikachu" (count > 0).	Prints "Card added." and the card moves from collection to binder.	Prints "Card added." and the card moves from collection to binder.	P
handleAddCardToBinder	2	Fails to add a card with zero available count.	User selects "Trade Binder", then selects "Mewtwo" (count = 0).	Prints "Failed to add card..." and no state is changed.	Prints "Failed to add card..." and no state is changed.	P
handleRemoveCardFromBin	1	Successfully removes a card from a binder.	User selects "Trade Binder", then selects an existing card "Pikachu" from it.	Prints "Card returned to collection." and the card moves from binder to collection.	Prints "Card returned to collection." and the card moves from binder to collection.	P
	1	Completes a valid trade.	User selects a binder, selects "Pikachu" to trade away, enters valid details for a new card "Charmander".	Prints "Trade successful!" and "Pikachu" is gone, "Charmander" is now in the binder.	Prints "Trade successful!" and "Pikachu" is gone, "Charmander" is now in the binder.	P
	2	Cancels an unfair trade upon user prompt.	Trade results in value difference >= $1.00. User enters "no" when prompted to proceed.	Prints "Trade canceled." and no state is changed.	Prints "Trade canceled." and no state is changed.	P
handleTradeFromBinder	3	Fails if incoming card details are invalid.	User enters a blank name for the incoming card.	Prints "Incoming card name cannot be blank." and the trade is aborted.	Prints "Incoming card name cannot be blank." and the trade is aborted.	P
handleCreateDeck	1	Successfully creates a new deck.	User enters a valid, unique name "Main Deck".	Prints "Deck created successfully." and the deck is added to DeckManager.	Prints "Deck created successfully." and the deck is added to DeckManager.	P
	1	Fails to add a duplicate card to a deck.	User selects "Main Deck" (already contains "Pikachu"), then selects "Pikachu" to add.	Prints "Failed to add card. (..., or duplicate card)". State is unchanged.	Prints "Failed to add card. (..., or duplicate card)". State is unchanged.	P
handleAddCardToDeck	2	Successfully adds a unique card to a deck.	User selects "Main Deck", then selects "Squirtle" (which is not in the deck).	Prints "Card added." and the card moves from collection to deck.	Prints "Card added." and the card moves from collection to deck.	P
handleRemoveCardFromDe	1	Successfully removes a card from a deck.	User selects "Main Deck", then a card from its list.	Prints "Card removed and returned to collection."	Prints "Card removed and returned to collection."	P
handleDeleteDeck	1	Successfully deletes a deck after confirmation.	User selects "Main Deck" and confirms with "yes".	Prints "Deck deleted."	Prints "Deck deleted."	P
	1	Returns the correct Card object on valid selection.	User selects card #2 from the main collection list.	The Card object corresponding to the second item in the sorted list is returned.	The Card object corresponding to the second item in the sorted list is returned.	P
	2	Returns null when user chooses to go back.	User selects option "0".	null is returned.	null is returned.	P
selectCardFromCollection	3	Handles an empty collection gracefully.	Method is called when no cards exist.	Prints "Collection is empty..." and returns null.	Prints "Collection is empty..." and returns null.	P
	1	Returns the correct Card from a provided list.	A list of 3 cards is provided. User selects #3.	The third Card object from the sorted list is returned.	The third Card object from the sorted list is returned.	P
selectCardFromList	2	Returns null if the provided list is empty.	An empty ArrayList is provided.	Prints "There are no cards to select." and returns null.	Prints "There are no cards to select." and returns null.	P
	1	Returns the correct Binder on valid selection.	User selects binder #1 from the list.	The Binder object for the first item in the sorted list is returned.	The Binder object for the first item in the sorted list is returned.	P
selectBinder	2	Returns null when no binders exist.	Method is called when binder list is empty.	Prints "No binders exist." and returns null.	Prints "No binders exist." and returns null.	P

selectDeck	1	Returns the correct Deck on valid selection.	User selects deck #2 from the list.	The Deck object for the second item in the sorted list is returned.	The Deck object for the second item in the sorted list is returned.	P
	2	Returns null when no decks exist.	Method is called when deck list is empty.	Prints "No decks exist." and returns null.	Prints "No decks exist." and returns null.	P
findCardIndexInList	1	Returns the correct index for an existing card.	List contains ["A", "B", "C"]. Search for "B".	1	1	P
	2	Returns -1 for a non-existent card.	List contains ["A", "B", "C"]. Search for "D".	-1	-1	P

Class: Menu

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
runMainMenu	1	Loops and waits for input, correctly delegates to a sub-menu.	User enters "1".	The Collection Menu is displayed.	The Collection Menu is displayed.	P
	2	Exits the loop when user enters "0".	User enters "0".	The runMainMenu method finishes execution.	The runMainMenu method finishes execution.	P
runCollectionMenu	1	Loops and correctly delegates to a handler method.	User enters "2".	The handler.handleAddNewCard() method is called.	The handler.handleAddNewCard() method is called.	P
runBinderMenu	1	Loops and correctly delegates to a handler method.	User enters "2".	The handler.handleCreateBinder() method is called.	The handler.handleCreateBinder() method is called.	P
runDeckMenu	1	Loops and correctly delegates to a handler method.	User enters "2".	The handler.handleCreateDeck() method is called.	The handler.handleCreateDeck() method is called.	P