Class: MainController

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
initMenu	1	Navigate to Pokemon menu	User input: 1	Pokemon menu displayed	Pokemon menu displayed	P
	2	Navigate to Moves menu	User input: 2	Moves menu displayed	Moves menu displayed	P
	3	Navigate to Items menu	User input: 3	Items menu displayed	Items menu displayed	P
	4	Exit application	User input: 4	Application exits	Application exits	P
	5	Invalid menu choice (negative)	User input: 9	"Out of Range!" message, menu redisplayed	"Out of Range!" message, menu redisplayed	P
	6	Non-numeric input (negative)	User input: "abc"	"Invalid Input!" message, menu redisplayed	"Invalid Input!" message, menu redisplayed	P
initPokemonMenu	7	Valid Pokemon menu navigation	User inputs: 1-6	Appropriate Pokemon submenu actions	Appropriate Pokemon submenu actions	P
	8	Invalid Pokemon menu choice (negative)	User input: 7	"Out of Range!" message, menu redisplayed	"Out of Range!" message, menu redisplayed	P
initMovesMenu	9	Valid Moves menu navigation	User inputs: 1-6	Appropriate Moves submenu actions	Appropriate Moves submenu actions	P
	10	Invalid Moves menu choice (negative)	User input: 0	"Out of Range!" message, menu redisplayed	"Out of Range!" message, menu redisplayed	P
initItemsMenu	11	Valid Items menu navigation	User inputs: 1-3	Appropriate Items submenu actions	Appropriate Items submenu actions	P
	12	Invalid Items menu choice (negative)	User input: 4	"Out of Range!" message, menu redisplayed	"Out of Range!" message, menu redisplayed	P

Class: ConsoleView

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
promptInt	1	Valid integer input	User enters "5"	Returns 5	Returns 5	P
	2	Invalid input then valid (negative)	User enters "abc" then "5"	"Invalid input!" message, then returns 5	"Invalid input!" message, then returns 5	P
	3	Negative integer	User enters "-10"	Returns -10	Returns -10	P
promptIntRange	4	Valid range input	Input: "3", range: 1-5	Returns 3	Returns 3	P
	5	Out of range input (negative)	Input: "10", range: 1-5	"Out of Range!" message, re-prompt	"Out of Range!" message, re-prompt	P
	6	Invalid then valid input	Input: "abc" then "3", range: 1-5	Error messages, then returns 3	Error messages, then returns 3	P

Class: PokemonController

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
savePokemonEntries	1	Save valid Pokemon list	List with 3 valid Pokemon objects	File saved successfully, success message	File saved successfully, success message	P
	2	Save empty Pokemon list (boundary)	Empty Pokemon list	Empty file created, success message	Empty file created, success message	P
	3	Save list with null Pokemon	List with 2 valid Pokemon and 1 null	File with 2 Pokemon entries, null skipped	File with 2 Pokemon entries, null skipped	P
loadPokemonEntries	4	Load valid Pokemon file	Existing valid Pokedex.txt	Pokemon list populated with file data	Pokemon list populated with file data	P
	5	Load non-existent file (negative)	Non-existent file path	Error handling, empty list or error message	Error handling, empty list or error message	P
	6	Load corrupted file (negative)	File with invalid/incomplete data	Exception caught, error message displayed	Exception caught, error message displayed	P
viewAllPokemon	7	View populated Pokemon list	List with 3 valid Pokemon	All Pokemon displayed via PokemonView	All Pokemon displayed via PokemonView	P
	8	View empty Pokemon list (boundary)	Empty Pokemon list	"No Pokemon Entries." message	"No Pokemon Entries." message	P
searchPokemonMenu	9	Search by valid attribute and key	attribute="name", key="Pikachu"	Matching Pokemon displayed with count	Matching Pokemon displayed with count	P
	10	Search with no matches	attribute="name", key="NonExistent"	"No Pokemon found" message	"No Pokemon found" message	P
	11	Search with invalid attribute initially	First: "invalid", Second: "name", key="Pikachu"	Error message, re-prompt, then valid results	Error message, re-prompt, then valid results	P

Class: PokemonView

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
viewPokemon	1	Display Pokemon with all valid data	Complete Pokemon object with all fields	Formatted Pokemon details with stats and evolution	Formatted Pokemon details with stats and evolution	P
	2	Display Pokemon with null type2	Pokemon object with type2 = null	Pokemon details with single type displayed	Pokemon details with single type displayed	P
	3	Display Pokemon with null heldItem	Pokemon object with heldItem = null	Pokemon details with "null" for held item	Pokemon details with "null" for held item	P
viewAllPokemon	4	Display list with multiple Pokemon	ArrayList with 3 valid Pokemon objects	Formatted list showing #, name, and type(s) for each	Formatted list showing #, name, and type(s) for each	P
	5	Display empty Pokemon list (boundary)	Empty ArrayList<Pokemon>	"---" + "No Pokemon Entries." + "---"	"---" + "No Pokemon Entries." + "---"	P
	6	Display list with null Pokemon	ArrayList with 2 valid Pokemon and 1 null	List showing only 2 valid Pokemon, null skipped	List showing only 2 valid Pokemon, null skipped	P
	7	Display list with all null Pokemon (boundary)	ArrayList with only null entries	"---" + "No Pokemon Entries." + "---"	"---" + "No Pokemon Entries." + "---"	P
	8	Display Pokemon with dual types	Pokemon with both type1 and type2	Pokemon displayed as "Name Type1/Type2"	Pokemon displayed as "Name Type1/Type2"	P
	9	Display Pokemon with single type	Pokemon with only type1	Pokemon displayed as "Name Type1"	Pokemon displayed as "Name Type1"	P
viewMoveSet	10	Display Pokemon with full moveSet	Pokemon with 4 valid moves	Numbered list of 4 moves (1] Move1, 2] Move2, etc.)	Numbered list of 4 moves (1] Move1, 2] Move2, etc.)	P
	11	Display Pokemon with partial moveSet	Pokemon with 2 moves, 2 nulls	Numbered list of 2 moves only, nulls skipped	Numbered list of 2 moves only, nulls skipped	P
	12	Display Pokemon with empty moveSet (boundary)	Pokemon with all null moves	No moves displayed (all skipped)	No moves displayed (all skipped)	P
	13	Display Pokemon with single move	Pokemon with 1 move, 3 nulls	Single numbered move displayed	Single numbered move displayed	P

Class: PokemonFileHandler

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
save	1	Save valid Pokemon list	ArrayList with 3 valid Pokemon objects	File created with Pokemon data, "Successfully Saved!" message	File created with Pokemon data, "Successfully Saved!" message	P
	2	Save empty Pokemon list (boundary)	Empty ArrayList<Pokemon>	Empty file created, "Successfully Saved!" message	Empty file created, "Successfully Saved!" message	P

	3	Save list with null Pokemon	ArrayList with 2 valid Pokemon and 1 null	File with only 2 Pokemon entries, null skipped	File with only 2 Pokemon entries, null skipped	P
	4	Save Pokemon with null moveSet	Pokemon with null moves in moveSet array	File created with "N/As" for null moves	File created with "N/As" for null moves	P
	5	Save Pokemon with null optional fields	Pokemon with null type2, heldItem	File created with "N/As" for null fields	File created with "N/As" for null fields	P
	6	Save to invalid file path (negative)	Valid Pokemon list, invalid file path	IOException caught, "An error occurred." message	IOException caught, "An error occurred." message	P
	7	Save null Pokemon list (boundary)	null ArrayList	NullPointerException or handled gracefully	NullPointerException or handled gracefully	P
load	8	Load valid Pokemon file	Existing "model/db/Pokedex.txt" with valid data	ArrayList<Pokemon> with correct objects, "Successfully Loaded!" message	ArrayList<Pokemon> with correct objects, "Successfully Loaded!" message	P
	9	Load from non-existent file (negative)	Non-existent file path	Exception caught, "An error occurred." message, empty ArrayList returned	Exception caught, "An error occurred." message, empty ArrayList returned	P
	10	Load from empty file (boundary)	Empty "model/db/Pokedex.txt" file	Empty ArrayList<Pokemon>, "Successfully Loaded!" message	Empty ArrayList<Pokemon>, "Successfully Loaded!" message	P

Class: PokemonManagement

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
searchPokemon	1	Search by exact Pokemon name	attribute="name", key="Pikachu"	ArrayList containing Pokemon with name "Pikachu"	ArrayList containing Pokemon with name "Pikachu"	P
	2	Search by partial Pokemon name	attribute="name", key="Pika"	ArrayList containing Pokemon with names containing "Pika"	ArrayList containing Pokemon with names containing "Pika"	P
	3	Search by type1	attribute="type", key="Electric"	ArrayList containing Pokemon with Electric as type1	ArrayList containing Pokemon with Electric as type1	P
	4	Search by type2	attribute="type", key="Flying"	ArrayList containing Pokemon with Flying as type1 or type2	ArrayList containing Pokemon with Flying as type1 or type2	P
	5	Search by exact Pokedex number	attribute="pokedex", key="25"	ArrayList containing Pokemon with Pokedex number 25	ArrayList containing Pokemon with Pokedex number 25	P
	6	Search with case insensitive input	attribute="TYPE", key="electric"	Same results as exact case match	Same results as exact case match	P
	7	Search with non-existent data (negative)	attribute="name", key="NonExistent"	Empty ArrayList	Empty ArrayList	P
	8	Search with invalid attribute (negative)	attribute="invalid", key="test"	Empty ArrayList	Empty ArrayList	P
	9	Search in empty Pokemon list (boundary)	attribute="name", key="Pikachu"	Empty ArrayList	Empty ArrayList	P
	10	Search with null Pokemon in list	attribute="name", key="Pikachu"	ArrayList with matching non-null Pokemon only	ArrayList with matching non-null Pokemon only	P
addPokemon	11	Add valid Pokemon	Valid Pokemon object	Pokemon added to list, list size increased	Pokemon added to list, list size increased	P
	12	Add null Pokemon (boundary)	null Pokemon object	null added to list (based on implementation)	null added to list (based on implementation)	P
setPokemonList	13	Set valid Pokemon list	ArrayList with 3 valid Pokemon	Pokemon list updated with 3 Pokemon	Pokemon list updated with 3 Pokemon	P
	14	Set list with null Pokemon	ArrayList with 2 valid Pokemon and 1 null	List contains only 2 valid Pokemon, null skipped	List contains only 2 valid Pokemon, null skipped	P
	15	Set empty list (boundary)	Empty ArrayList	Pokemon list cleared	Pokemon list cleared	P
isDupePokedexNum	16	Check existing Pokedex number	num=25 (exists in list)	TRUE	TRUE	P
	17	Check non-existing Pokedex number	num=999 (doesn't exist)	FALSE	FALSE	P
	18	Check in empty list (boundary)	num=25, empty Pokemon list	FALSE	FALSE	P
	19	Check with null Pokemon in list	num=25, list with null entries	Correct boolean result ignoring nulls	Correct boolean result ignoring nulls	P
isDupeName	20	Check existing Pokemon name	name="Pikachu" (exists in list)	TRUE	TRUE	P
	21	Check non-existing Pokemon name	name="NonExistent"	FALSE	FALSE	P
	22	Check with case insensitive match	name="PIKACHU" (exists as "Pikachu")	TRUE	TRUE	P
	23	Check in empty list (boundary)	name="Pikachu", empty list	FALSE	FALSE	P

Class: ItemsController

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
viewAllItems	1	View items with populated list	Items list with 3 valid items	All items displayed via ItemsView	All items displayed via ItemsView	P
	2	View items with empty list (boundary)	Empty items list	Empty display or "no items" message	Empty display or "no items" message	P
	3	View items with null items in list	List with 2 valid items and 1 null	Only valid items displayed	Only valid items displayed	P
searchItem	4	Search by valid attribute and key	attribute="name", key="Potion"	Matching items displayed with count message	Matching items displayed with count message	P
	5	Search with no matches	attribute="name", key="NonExistent"	"No items found" message	"No items found" message	P
	6	Search with invalid attribute initially	First input: "invalid", Second: "name", key=" Potion"	Error message, re-prompt, then valid search results	Error message, re-prompt, then valid search results	P
	7	Search by category	attribute="category", key="Medicine"	Items in Medicine category displayed	Items in Medicine category displayed	P
	8	Search by keyword	attribute="keyword", key="heal"	Items with "heal" in description/effects displayed	Items with "heal" in description/effects displayed	P
	9	Search with case insensitive attribute	attribute="NAME", key="potion"	Same results as exact case match	Same results as exact case match	P
	10	Search with empty key (boundary)	attribute="name", key=""	All items displayed (empty string matches all)	All items displayed (empty string matches all)	P

Class: ItemsManagement

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
searchItems	1	Search by exact item name	attribute="name", key="Potion"	ArrayList containing Items with name "Potion"	ArrayList containing Items with name "Potion"	P
	2	Search by partial item name	attribute="name", key="pot"	ArrayList containing all Items with names containing "pot"	ArrayList containing all Items with names containing "pot"	P
	3	Search by item category	attribute="category", key="Medicine"	ArrayList containing all Items in "Medicine" category	ArrayList containing all Items in "Medicine" category	P
	4	Search by keyword in description	attribute="keyword", key="heal"	ArrayList containing Items with "heal" in description or effects	ArrayList containing Items with "heal" in description or effects	P

	5	Search by keyword in effects	attribute="keyword", key="restore"	ArrayList containing Items with "restore" in description or effects	ArrayList containing Items with "restore" in description or effects	P
	6	Search with case insensitive input	attribute="NAME", key="POTION"	Same results as lowercase search	Same results as lowercase search	P
	7	Search with non-existent key (negative)	attribute="name", key="NonExistent"	Empty ArrayList	Empty ArrayList	P
	8	Search with invalid attribute (negative)	attribute="invalid", key="test"	Empty ArrayList	Empty ArrayList	P
	9	Search in empty item list (boundary)	attribute="name", key="Potion"	Empty ArrayList	Empty ArrayList	P
	10	Search with null items in list	attribute="name", key="Potion"	ArrayList with matching non-null items only	ArrayList with matching non-null items only	P

Class: ItemsView

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
viewItems	1	Display items with all valid data	ArrayList with 2 complete Items	Formatted item display with all details	Formatted item display with all details	P
	2	Display items with null entries	List with null Items	Only non-null items displayed	Only non-null items displayed	P
	3	Display empty list (boundary)	Empty ArrayList	No output or minimal display	No output or minimal display	P

Class: ItemsFileHandler

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
load	1	Load valid Items file	Existing "model/db/Items.txt" with valid data	ArrayList<Items> with correct item objects	ArrayList<Items> with correct item objects	P
	2	Load from non-existent file (negative)	Non-existent file path	IOException caught, "An error occurred." message, empty ArrayList returned	IOException caught, "An error occurred." message, empty ArrayList returned	P
	3	Load from empty file (boundary)	Empty "model/db/Items.txt" file	Empty ArrayList<Items>	Empty ArrayList<Items>	P
	4	Load file with malformed data (negative)	File with incomplete/invalid item data	Exception caught, "An error occurred." message	Exception caught, "An error occurred." message	P
	5	Load file with N/As tokens	File containing "N/As" values	ArrayList with Items objects having null values where N/As found	ArrayList with Items objects having null values where N/As found	P
	6	Load file with invalid double data (negative)	File with non-numeric values for price fields	NumberFormatException caught, "An error occurred." message	NumberFormatException caught, "An error occurred." message	P
	7	Load file with missing tokens (negative)	File with insufficient data per line	ArrayIndexOutOfBoundsException caught, error handling	ArrayIndexOutOfBoundsException caught, error handling	P
	8	Load file with negative prices	File with negative price values	ArrayList with Items having negative price values	ArrayList with Items having negative price values	P
	9	Load file with zero prices (boundary)	File with price values = 0.0	ArrayList with Items having zero price values	ArrayList with Items having zero price values	P
	10	Load file with large price values	File with prices > 999999.99	ArrayList with Items having large price values	ArrayList with Items having large price values	P

Class: MovesController

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
searchMoves	1	Search with valid attribute and keyword	attribute="name", keyword="Tackle"	Matching moves displayed with count message	Matching moves displayed with count message	P
	2	Search with no matches	attribute="name", keyword="NonExistent"	"No moves found" message	"No moves found" message	P
	3	Search with invalid attribute initially	First input: "invalid", Second: "name", keyword="Tackle"	Error message, re-prompt, then valid search results	Error message, re-prompt, then valid search results	P
	4	Search by classification	attribute="classification", keyword=" Physical"	Moves with Physical classification displayed	Moves with Physical classification displayed	P
	5	Search by type	attribute="type", keyword="Fire"	Moves with Fire type displayed	Moves with Fire type displayed	P
	6	Search with case insensitive attribute	attribute="NAME", keyword="tackle"	Same results as exact case match	Same results as exact case match	P
	7	Search with empty keyword (boundary)	attribute="name", keyword=""	All moves displayed (empty string matches all)	All moves displayed (empty string matches all)	P
	8	Search in empty moves list (boundary)	attribute="name", keyword="Tackle"	"No moves found" message	"No moves found" message	P
addMoves	9	Add valid move with all fields	name="Thunder", type="Electric", classification="TM", desc="Electric attack"	Move added successfully, success message	Move added successfully, success message	P
	10	Add move with empty name (negative)	name="", type="Electric", classification=" TM", desc="Attack"	"Invalid input!" error message	"Invalid input!" error message	P
	11	Add move with empty type (negative)	name="Thunder", type="", classification=" TM", desc="Attack"	"Invalid input!" error message	"Invalid input!" error message	P
	12	Add move with empty classification (negative)	name="Thunder", type="Electric", classification="", desc="Attack"	"Invalid input!" error message	"Invalid input!" error message	P
	13	Add move with empty description	name="Thunder", type="Electric", classification="TM", desc=""	Move added successfully (description can be empty)	Move added successfully (description can be empty)	P
	14	Add move with all fields empty (negative)	name="", type="", classification="", desc=""	"Invalid input!" error message	"Invalid input!" error message	P
viewMoves	15	View moves with populated list	Moves list with 3 valid moves	All moves displayed via MovesView	All moves displayed via MovesView	P
	16	View moves with empty list (boundary)	Empty moves list	"No moves in list." message	"No moves in list." message	P
saveMoves	17	Save current moves list	Moves list with valid data	File saved successfully via fileHandler	File saved successfully via fileHandler	P
	18	Save empty moves list (boundary)	Empty moves list	Empty file saved successfully	Empty file saved successfully	P
loadMoves	19	Load moves from file	Existing valid moves file	Moves list populated with file data	Moves list populated with file data	P
	20	Load from non-existent file (negative)	Non-existent moves file	Error handling via fileHandler	Error handling via fileHandler	P

Class: MovesView

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
displayMoves	1	Display list with multiple moves	ArrayList with 3 valid Moves objects	Header + formatted table with 3 move entries	Header + formatted table with 3 move entries	P
	2	Display empty moves list (boundary)	Empty ArrayList<Moves>	"=== MOVE LIST ===" + "No moves in list."	"=== MOVE LIST ===" + "No moves in list."	P
	3	Display list with null moves	ArrayList with 2 valid moves and 1 null	Header + table with 2 moves (null skipped via displayMove)	Header + table with 2 moves (null skipped via displayMove)	P

	4	Display single move	ArrayList with 1 valid Move	Header + formatted table with 1 move entry	Header + formatted table with 1 move entry	P
displayMove	5	Display valid move	Valid Moves object with all fields	Formatted line with name, type, classification, description	Formatted line with name, type, classification, description	P
	6	Display move with null fields (boundary)	Moves object with some null fields	Formatted line with "null" displayed for null fields	Formatted line with "null" displayed for null fields	P
	7	Display move with long description	Moves object with 60+ character description	Formatted line with description (may wrap or truncate)	Formatted line with description (may wrap or truncate)	P

Class: MovesManagement

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
setMoveList	1	Set valid move list	ArrayList with 3 valid Moves objects	moves list updated with 3 moves, previous list cleared	moves list updated with 3 moves, previous list cleared	P
	2	Set list with null moves	ArrayList with 2 valid moves and 1 null	moves list contains only 2 valid moves, null skipped	moves list contains only 2 valid moves, null skipped	P
	3	Set empty list (boundary)	Empty ArrayList<Moves>	moves list cleared and remains empty	moves list cleared and remains empty	P
	4	Set list replacing existing moves	New ArrayList with 2 moves, existing list has 3	Old list cleared, new list with 2 moves set	Old list cleared, new list with 2 moves set	P
	5	Set list with all null moves (boundary)	ArrayList with 3 null Moves objects	moves list cleared and remains empty (all nulls skipped)	moves list cleared and remains empty (all nulls skipped)	P
	6	Set null list (boundary)	null ArrayList	NullPointerException thrown	NullPointerException thrown	P
	7	Set large move list	ArrayList with 100 valid Moves objects	moves list updated with all 100 moves	moves list updated with all 100 moves	P
searchMoves	8	Search by exact move name	attribute="name", keyword="Tackle"	ArrayList containing move with name "Tackle"	ArrayList containing move with name "Tackle"	P
	9	Search by partial move name	attribute="name", keyword="tack"	ArrayList containing moves with names containing "tack"	ArrayList containing moves with names containing "tack"	P
	10	Search by move name case insensitive	attribute="name", keyword="TACKLE"	ArrayList containing move with name "Tackle"	ArrayList containing move with name "Tackle"	P
	11	Search by exact classification	attribute="classification", keyword=" Physical"	ArrayList containing moves with "Physical" classification	ArrayList containing moves with "Physical" classification	P
	12	Search by partial classification	attribute="classification", keyword="phys"	ArrayList containing moves with classifications containing "phys"	ArrayList containing moves with classifications containing "phys"	P
	13	Search by classification case insensitive	attribute="classification", keyword=" PHYSICAL"	ArrayList containing moves with "Physical" classification	ArrayList containing moves with "Physical" classification	P
	14	Search by exact move type	attribute="type", keyword="Fire"	ArrayList containing moves with "Fire" type	ArrayList containing moves with "Fire" type	P
	15	Search by partial move type	attribute="type", keyword="fir"	ArrayList containing moves with types containing "fir"	ArrayList containing moves with types containing "fir"	P
	16	Search by type case insensitive	attribute="type", keyword="FIRE"	ArrayList containing moves with "Fire" type	ArrayList containing moves with "Fire" type	P
	17	Search with non-existent keyword (negative)	attribute="name", keyword=" NonExistentMove"	Empty ArrayList	Empty ArrayList	P
	18	Search with invalid attribute (negative)	attribute="invalid", keyword="test"	Empty ArrayList (no case matches)	Empty ArrayList (no case matches)	P
	19	Search with empty keyword (boundary)	attribute="name", keyword=""	ArrayList containing all moves (empty string matches all)	ArrayList containing all moves (empty string matches all)	P
	20	Search in empty moves list (boundary)	attribute="name", keyword="Tackle", empty moves list	Empty ArrayList	Empty ArrayList	P
	21	Search with null keyword (boundary)	attribute="name", keyword=null	NullPointerException thrown	NullPointerException thrown	P
	22	Search with null attribute (boundary)	attribute=null, keyword="Tackle"	NullPointerException thrown	NullPointerException thrown	P
	23	Search multiple matches	attribute="type", keyword="Normal", 3 Normal-type moves in list	ArrayList containing all 3 Normal-type moves	ArrayList containing all 3 Normal-type moves	P
	24	Search with special characters in keyword	attribute="name", keyword="Thunder-Punch"	ArrayList containing moves with names containing "Thunder-Punch"	ArrayList containing moves with names containing "Thunder-Punch"	P
	25	Search with numeric characters in keyword	attribute="classification", keyword="TM25"	ArrayList containing moves with classifications containing "TM25"	ArrayList containing moves with classifications containing "TM25"	P
addMove	26	Add valid move to empty list	Valid Moves object	Move added to list, list size = 1	Move added to list, list size = 1	P
	27	Add valid move to existing list	Valid Moves object, list already has 2 moves	Move added to list, list size = 3	Move added to list, list size = 3	P
	28	Add null move (boundary)	null Moves object	null added to list, list size increased by 1	null added to list, list size increased by 1	P
	29	Add multiple moves sequentially	3 different valid Moves objects added one by one	All 3 moves added to list, list size = 3	All 3 moves added to list, list size = 3	P
	30	Add duplicate move	Same Moves object added twice	Both instances added to list (duplicates allowed)	Both instances added to list (duplicates allowed)	P
	31	Add move with null fields	Moves object with null name, type, etc.	Move added to list regardless of null fields	Move added to list regardless of null fields	P
	32	Add move with empty string fields	Moves object with empty strings for fields	Move added to list with empty string fields	Move added to list with empty string fields	P
getMoves	33	Get moves from populated list	List with 3 valid moves	Returns ArrayList with 3 moves	Returns ArrayList with 3 moves	P
	34	Get moves from empty list (boundary)	Empty moves list	Returns empty ArrayList	Returns empty ArrayList	P
	35	Get moves after modifications	List modified by add/set operations	Returns current state of moves list	Returns current state of moves list	P
	36	Get moves reference test	Check if returned list is same reference	Original list and returned list should be same reference	Original list and returned list should be same reference	P

Class: MovesFileHandler

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
save	1	Save valid list of moves to file	ArrayList with 3 valid Moves objects	File created with moves data, "Successfully saved!" message	File created with moves data, "Successfully saved!" message	P
	2	Save empty list of moves	Empty ArrayList<Moves>	Empty file created, "Successfully saved!" message	Empty file created, "Successfully saved!" message	P
	3	Save list with null moves (boundary)	ArrayList with 2 valid moves and 1 null	File with only 2 moves, null skipped, "Successfully saved!" message	File with only 2 moves, null skipped, "Successfully saved!" message	P
	4	Save to invalid file path (negative)	Valid moves list, invalid file path	IOException caught, "An error occurred." message	IOException caught, "An error occurred." message	P

	5	Save null list (boundary)	null ArrayList	NullPointerException or handled gracefully	NullPointerException or handled gracefully	P
load	6	Load valid moves file	Existing "model/db/Moves.txt" with valid data	ArrayList<Moves> with correct move objects, "Successfully loaded!" message	ArrayList<Moves> with correct move objects, "Successfully loaded!" message	P
	7	Load from non-existent file (negative)	Non-existent file path	IOException caught, "An error occurred." message, empty ArrayList returned	IOException caught, "An error occurred." message, empty ArrayList returned	P
	8	Load from empty file (boundary)	Empty "model/db/Moves.txt" file	Empty ArrayList<Moves>, "Successfully loaded!" message	Empty ArrayList<Moves>, "Successfully loaded!" message	P
	9	Load file with malformed data (negative)	File with incomplete/invalid move data	Exception caught, "An error occurred." message	Exception caught, "An error occurred." message	P
	10	Load file with N/As tokens	File containing "N/As" values	ArrayList with Moves objects having null values where N/As found	ArrayList with Moves objects having null values where N/As found	P

Class: FileHelper

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
safe	1	Convert valid string	"Hello"	"Hello|"	"Hello|"	P
	2	Convert valid integer	123	"123|"	"123|"	P
	3	Convert valid double	45.67	"45.67|"	"45.67|"	P
	4	Convert null value (boundary)	null	"N/As|"	"N/As|"	P
	5	Convert empty string	""	"|"	"|"	P
fromSafe	6	Convert regular string	"Hello"	"Hello"	"Hello"	P
	7	Convert N/As string	"N/As"	null	null	P
	8	Convert empty string (boundary)	""	""	""	P
	9	Convert null input (boundary)	null	null or NullPointerException	null or NullPointerException	P

Class: InputHelper

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
checkNA	1	Check valid non-N/A string	"Pikachu"	"Pikachu"	"Pikachu"	P
	2	Check lowercase "n/a"	"n/a"	null	null	P
	3	Check uppercase "N/A"	"N/A"	null	null	P
	4	Check mixed case "N/a"	"N/a"	null	null	P
	5	Check lowercase "na"	"na"	null	null	P
	6	Check uppercase "NA"	"NA"	null	null	P
	7	Check mixed case "Na"	"Na"	null	null	P
	8	Check empty string (boundary)	""	null	null	P
	9	Check whitespace string	" "	" " (not considered N/A)	" " (not considered N/A)	P
	10	Check null input (boundary)	null	NullPointerException or handled gracefully	NullPointerException or handled gracefully	P
	11	Check string with N/A substring	"Not N/A"	"Not N/A" (not exact match)	"Not N/A" (not exact match)	P
	12	Check numeric string	"123"	"123"	"123"	P

Class: Trainer

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
Constructor (with ID)	1	Create trainer with valid parameters	ID: 100001, name: "Ash", birthDate: "1990-01-01	Trainer object created with all fields set, lineup initialized as empty array, pokemonBox as empty ArrayL		P
	2	Create trainer with null name	ID: 100002, name: null, other valid params	Trainer object created with name = null, other fields set correctly		P
Constructor (without ID)	3	Create new trainer with auto-generated ID	name: "Misty", birthDate: "1992-05-05", sex: "Fe	Trainer created with auto-generated ID (starting from 100000), money set to 1000000 by default		P
addMoney	4	Add positive amount to money	trainer with money: 1000.0, amount: 500.0	money becomes 1500.0		P
	5	Add zero amount	trainer with money: 1000.0, amount: 0.0	money remains 1000.0		P
	6	Add negative amount	trainer with money: 1000.0, amount: -200.0	money becomes 800.0		P
deductMoney	7	Deduct valid amount (sufficient funds)	trainer with money: 1000.0, amount: 300.0	returns true, money becomes 700.0		P
	8	Deduct amount equal to current money	trainer with money: 1000.0, amount: 1000.0	returns true, money becomes 0.0		P
	9	Deduct amount exceeding current money	trainer with money: 500.0, amount: 600.0	returns false, money remains 500.0		P
setPokemonLineup	10	Set valid Pokemon array	Pokemon array with 3 valid Pokemon objects	pokemonLineup updated with new array		P
	11	Set null array	null Pokemon array	pokemonLineup set to null		P

Class: TrainerManagement

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
addTrainer	1	Add valid trainer to empty list	Valid Trainer object	trainer added to ArrayList, size becomes 1		P
	2	Add multiple trainers	3 different valid Trainer objects	all trainers added, size becomes 3		P
	3	Add null trainer	null Trainer object	null added to list (no validation)		P
addToStorage	4	Add Pokemon to empty storage	trainer with empty pokemonBox, valid Pokemon	Pokemon added to trainer's pokemonBox, size becomes 1		P
	5	Add Pokemon to existing storage	trainer with 2 Pokemon in storage, new Pokemon	Pokemon added, storage size becomes 3		P
	6	Add null Pokemon to storage	trainer, null Pokemon	null added to storage		P
canAddPokemon	7	Check trainer with empty lineup	trainer with all lineup slots null	returns true		P
	8	Check trainer with full lineup	trainer with all 6 lineup slots filled	returns false		P
	9	Check trainer with partially filled lineup	trainer with 3 Pokemon in lineup, 3 null slots	returns true		P
addPokemon	10	Add Pokemon to empty lineup	trainer with empty lineup, valid Pokemon	returns 1, Pokemon added to first slot, lineupCount becomes 1		P
	11	Add Pokemon to full lineup	trainer with 6 Pokemon in lineup, new Pokemon	returns 0, Pokemon added to storage instead		P
	12	Add null Pokemon	trainer, null Pokemon	returns -1, no changes made		P
	13	Add Pokemon when lineup has 5 Pokemon	trainer with 5 Pokemon, new Pokemon	returns 1, Pokemon added to 6th slot, lineupCount becomes 6		P
switchPokemon	14	Valid switch between lineup and storage	trainer, boxIndex: 0, lineupIndex: 2	returns true, Pokemon swapped between positions		P
	15	Invalid box index (negative)	trainer, boxIndex: -1, lineupIndex: 0	returns false, no changes made		P
	16	Invalid box index (too large)	trainer with 2 storage Pokemon, boxIndex: 5, line	returns false, no changes made		P

	17	Switch to empty lineup slot (lineupIndex: -1)	trainer with storage Pokemon, boxIndex: 0, lineup	returns true, Pokemon moved from storage to first empty lineup slot		P
releasePokemon	18	Release Pokemon from lineup	trainer, Pokemon object in lineup	Pokemon removed from lineup, lineup shifted left		P
	19	Release Pokemon from storage	trainer, Pokemon object in storage	Pokemon removed from storage ArrayList		P
	20	Release Pokemon not owned by trainer	trainer, Pokemon not in lineup or storage	no changes made to trainer		P
buyItem	21	Buy item with sufficient funds	trainer with 1000 money, item costing 500, quanti	returns positive value, money deducted, item added to inventory		P
	22	Buy item with insufficient funds	trainer with 100 money, item costing 500, quantity	returns -1, no changes made		P
	23	Buy item with zero quantity	trainer, valid item, quantity: 0	returns 0, no changes made		P
sellItem	24	Sell item trainer owns	trainer with item in inventory, valid quantity	returns true, money added, item quantity reduced in inventory		P
	25	Sell item trainer doesn't own	trainer without item, item to sell	returns false, no changes made		P
	26	Sell more items than owned	trainer with 2 of item, quantity to sell: 5	returns false, no changes made		P
useItem	27	Use valid item on compatible Pokemon	trainer, Pokemon, compatible item (e.g., Potion)	returns true, item effect applied, item consumed from inventory		P
	28	Use evolution stone on compatible Pokemon	trainer, Pokemon that can evolve, correct evolutio	returns true, Pokemon evolved, stone consumed		P
	29	Use item trainer doesn't have	trainer without item, Pokemon, item	returns false, no changes made		P
	30	Use item on incompatible Pokemon	trainer, Pokemon, incompatible item	returns false, item not consumed		P

Class: TrainerController

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
handleViewTrainer	1	View trainers when list is not empty	TrainerManagement with 3 trainers	ArrayList<String> with 3 trainer info strings returned		P
	2	View trainers when list is empty	TrainerManagement with no trainers	empty ArrayList<String> returned		P
handleSearchTrainer	3	Search by name with exact match	input: "Ash", attribute: "name"	ArrayList with matching trainer(s)		P
	4	Search by name with no match	input: "Unknown", attribute: "name"	empty ArrayList returned		P
	5	Search by ID with valid ID	input: "100001", attribute: "id"	ArrayList with matching trainer		P
	6	Search with invalid attribute	input: "Ash", attribute: "invalid"	empty ArrayList or error handling		P
handleAddPokemon	7	Add Pokemon to trainer with space	trainerId: "100001", dexNum: "25" (Pikachu)	Pokemon added to trainer's lineup, GUI updated		P
	8	Add Pokemon to trainer with full lineup	trainerId with 6 Pokemon, dexNum: "25"	Pokemon added to storage instead		P
	9	Add non-existent Pokemon	trainerId: "100001", dexNum: "999"	error message, no Pokemon added		P
handleAvailableItems	10	Get items from trainer with inventory	trainerId with items in inventory	ArrayList<String> with item descriptions		P
	11	Get items from trainer with empty inventory	trainerId with no items	empty ArrayList<String>		P
handleAvailablePokemonF	12	Get Pokemon from trainer with multiple Pokemon	trainerId with 3 lineup + 2 storage Pokemon	ArrayList with 4 Pokemon (excluding 1 lineup to prevent last Pokemon release)		P
	13	Get Pokemon from trainer with only 1 lineup Poke	trainerId with 1 lineup Pokemon only	ArrayList with only storage Pokemon (lineup Pokemon excluded)		P
	14	Get Pokemon from trainer with empty roster	trainerId with no Pokemon	empty ArrayList		P
	15					
	16					
	17					
	18					

Class: GUIUtils

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
createImageButton	1	Create button with valid image path	path: "assets/button.png", x: 100, y: 50	JButton created with image, positioned at (100,50), no borders/focus painting		P
	2	Create button with invalid image path	path: "invalid/path.png", x: 100, y: 50	JButton created but may have no image or default image		P
	3	Create button with negative coordinates	path: "assets/button.png", x: -50, y: -30	JButton created at negative position		P
createCenterImageButton	4	Create centered button with valid image	path: "assets/button.png", y: 200	JButton created, horizontally centered (x calculated as (640-width)/2), y: 200		P
	5	Create centered button with very wide image	path: "assets/wide_button.png", y: 100	JButton created, x may be negative if image wider than 640px		P
createText	6	Create text label with valid parameters	txt: "Hello World", x: 50, y: 100, w: 200, h: 30	JLabel created with text, positioned and sized correctly, red border visible		P
	7	Create text label with empty string	txt: "", x: 50, y: 100, w: 200, h: 30	JLabel created with no visible text, red border still visible		P
	8	Create text label with null text	txt: null, x: 50, y: 100, w: 200, h: 30	JLabel created, may show "null" or empty		P
createBanner	9	Create banner with valid image	path: "assets/banner.png", x: 0, y: 0	JLabel created with image at specified position		P
	10	Create banner with invalid image path	path: "nonexistent.png", x: 0, y: 0	JLabel created but may not display image		P
createCenterBanner	11	Create centered banner	path: "assets/title.png", y: 50	JLabel with image, horizontally centered, y: 50		P
	12	Create centered banner with very wide image	path: "assets/wide_title.png", y: 50	JLabel created, x may be negative if image wider than 640px		P
createTextField	13	Create text field with valid dimensions	x: 100, y: 150, w: 200, h: 25	JTextField created with red border, transparent background, positioned correctly		P
	14	Create text field with zero dimensions	x: 100, y: 150, w: 0, h: 0	JTextField created but not visible due to zero size		P
	15	Create text field with negative dimensions	x: 100, y: 150, w: -50, h: -25	JTextField created with unusual behavior due to negative dimensions		P

Class: MainGUI

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
showTrainer	1	Display trainer with valid data	String array with complete trainer info	Trainer screen displayed with all fields, buttons, and Pokemon lineup visible		P
	2	Display trainer with null values in info	String array with some null entries	Screen displayed with empty/default values for null entries		P
	3	Display trainer with empty Pokemon lineup	Trainer info with no Pokemon in lineup	Screen displayed with empty lineup buttons		P
showViewScreen	4	Display list with multiple items	ArrayList with 5 items, valid paths	Scrollable list displayed with all items as buttons		P
	5	Display empty list	Empty ArrayList, valid paths	Screen displayed with title and back button, no item buttons		P
	6	Display list with "release" backPath	ArrayList with items, backPath: "release"	List displayed with proper navigation back to trainer screen		P
setPrompt	7	Set prompt with valid message	prompt: "Pokemon released successfully!"	Message displayed in promptLabel with HTML formatting		P
	8	Set prompt with null message	prompt: null	promptLabel updated, may show "null" or empty		P
	9	Set prompt when promptLabel is null	any prompt message	No error, method handles null promptLabel gracefully		P