ENSF 380 Group #11 Group Project #2 Winter 2025

Personal Note (Ignore): Accepts New Instance of Class Objects and New Instance Array of class objects with the same attributes as the argument/parameters of their respective class constructors. Array of class objects, so there would be two instances of the class object therefore there will be two versions of the attributes for the Supply class so two instances of that class object will have two versions of its attributes in that array of class objects meaning that each element inside this array would represent one attribute for each instance of that class object within the array of class objects.

Testing The Public Member Methods Of Each Class And Exceptions For Each Class And Their Respectful Methods Assigned To Each Of Them And For Our Tests We Will Be Using The Example Sample Data From The Previous Group Project As Well As Some Other Examples For The Test Value(s) Section:

Class #1: DisasterVictim:

Test Name	Description	Test Value(s)
testDisasterVictimConstructor (Both Constructors are tested respectively as a constructor can be overloaded.)	The constructor for the DisasterVictim Class is responsible for acting as the template for each DisasterVictim Class object to be instantiated with the two or three argument/parameters for the overloaded constructor for both the two and three arguments/parameters include the following: 'firstName' and 'ENTRY_DATE' and 'dateOfBirth'. The getFirstName(), getEntryDate() and getDateOfBirth() are used to verify through retrieval of values and to check if the correct parameters/arguments were used when initialising a DisasterVictim class object.	1. "Teruya", "2025-01-18" 2. "John", "Doe", "2025-01-16"
testInvalidDisasterVictimConstructor	'DisasterVictim' constructor should have the proper String data type arguments/parameters following: ("firstName","ENTRY_DATE","dateOfBirth"). If the data type of the two or three parameters depending on which constructor is used happens to be anything other then 'String' then an 'IllegalArgumentException' will be flagged.	1. Teruya, 2025-01-18 (Non-String = Invalid) 2. John, Doe, 2025-01-16 (Non-String = Invalid)
testSetFirstName	Test to see if the 'setFirstName' method successfully updates and sets the 'firstName' String	"Freda"

	attribute to the given input value.	
testSetLastName	Test to see if the ‘setLastName’ method successfully updates and sets the ‘lastName’ String attribute to the given input value.	“McDonald”
testSetDateOfBirth	Test to see if the ‘setDateOfBirth’ method successfully updates and sets the ‘dateOfBirth’ String attribute to the given input value.	“1986-06-03” (String So Valid)
testInvalidSetDateOfBirth	‘setDateOfBirth’ method should have a String data type argument/parameter. Anything other than a String ‘dateOfBirth’ would cause the ‘IllegalArgumentException’ to be flagged.	1986-06-03 (Integer So Invalid)
testSetFamilyConnections	Test to see if the ‘setFamilyConnections’ method successfully updates and sets the attributes for the instance of the FamilyRelation class object within the array of FamilyRelation class objects represented by the argument/parameter variable ‘connections’ to the given input values. The ‘getFamilyConnections’ method of the type ‘FamilyConnection[ ]’ can be used to verify that the method has functioned correctly.	personOne: “Hans Massaquoi” personTwo: “Bertha Nikodijevic” relationshipTo: “Mother”
testSetMedicalRecords	Test to see if the ‘setMedicalRecords’ method successfully updated and sets the attributes for the instance of the MedicalRecord class object within the array of MedicalRecord class objects represented by the argument/parameter variable ‘records’ to the given input values. The ‘getMedicalRecords’ method of the type ‘MedicalRecord [ ]’ can be used to verify that the method has functioned correctly.	location: “University Of Calgary” treatmentDetails: “Pernicious Anemia” dateOfTreatment: “2025-01-19”

testSetPersonalBelonging	Test to see if the 'setPersonalBelongings' method successfully updated and sets the attributes for the instance of the Supply class object within the array of the Supply class objects represented by the argument/parameter variable 'belongings' to the given input values. The 'getPersonalBelongings' method of the type 'Supply [ ]' can be used to verify that the method has functioned correctly.	type: ["Toiletries Kit", "Clothes"] quantity: [1, 2]
testAddPersonalBelonging	Test to see if the 'addPersonalBelongings' method correctly adds a new supply by adding an instance of a Supply class as an object represented by the argument/parameter variable 'supply' of the Supply class type given the necessary input values. The 'getType' and 'getQuantity' methods of String and Integer data types respectively can be used to check if the 'addPersonalBelongings' method had correctly functioned as intended.	Type: "Canned Soup" Quantity: 4 Supply("Canned Soup", 4)
testRemovePersonalBelonging	Test to see if the 'removePersonalBelonging' method correctly removes a supply by removing an instance of a Supply class as an object represented by the argument/parameter variable 'unwantedSupply' of the Supply class type given the necessary input values. The 'getType' and 'getQuantity' methods of String and Integer data types respectively can be used to check if the 'removePersonalBelongings' method had correctly functioned as intended.	Type: "Canned Soup" Quantity: 2 Supply("Canned Soup", 2)
testRemoveFamilyConnection	Test to see if the 'removeFamilyConnection' method correctly removes a	personOne: Freda McDonald relationshipTo: Husband personTwo: Jo Bouillon

	family connection/relation by removing an instance of the FamilyRelation class as an object represented by the argument/parameter variable ‘exRelation’ of the FamilyRelation class type given the necessary input values. The ‘getPersonOne’, ‘getRelationshipTo’ and ‘getPersonTwo’ methods can be used to check if the ‘removeFamilyConnection’ method had correctly functioned as intended	FamilyRelation(“Freda McDonald, “Husband”, “Jo Bouillon”)
testAddFamilyConnection	Test to see if the ‘addFamilyConnection’ method correctly adds a new family connection/relation by adding an instance of the FamilyRelation class as an object represented by the argument/parameter variable ‘record’ of the FamilyRelation class type given the necessary input values. The ‘getPersonOne’, ‘getRelationshipTo’ and ‘getPersonTwo’ methods can be used to check if the ‘addFamilyConnection’ method had correctly functioned as intended	personOne: Freda McDonald relationshipTo: Husband personTwo: Jo Bouillon FamilyRelation(“Freda McDonald, “Husband”, “Jo Bouillon”)
testAddMedicalRecord	Test to see if the ‘addMedicalRecord’ method correctly adds a new medical record by adding an instance of the ‘MedicalRecord’ class as an object represented by the argument/parameter variable ‘record’ of the MedicalRecord class type given the necessary input values. The ‘getLocation’, ‘getTreatmentDetails’ and ‘getDateOfTreatment’ methods can be used to check if the ‘addMedicalRecord’ method had correctly functioned as intended.	location: “University of Calgary” treatmentDetails: “Ankle Sprained, Ice Packet And Physio Applied” dateOfTreatment: “2025-01-16” MedicalRecord(“University of Calgary”, “Ankle Sprained, Ice Packet And Physio Applied”, “2025-01-16”)
testSetComments	Test to see if the ‘setComments’ method correctly updates and sets the ‘comments’ attribute of	comment: “Freda McDonald (female, b. 1986-06-03), suffering from a twisted ankle

	the String data type to a comment about Disaster Victims. The ‘getComments’ method can be used to check if the ‘setComments’ method has correctly functioned as intended.	and light burns. She was not accompanied by her husband, Jo Bouillon, who is travelling and presumed safe. She is looking for her children, Teruya, Jari, and Luis Bouillon. She was issued a personal toiletries kit and two sets of clothes.”
testSetGender	Test to see if the ‘setGender’ method correctly updates and sets the ‘gender’ attribute of the String data type to a gender description of Disaster Victims. The ‘getGender’ method can be used to check if the ‘setGender’ method has correctly functioned as intended.	gender: “Female”
testInvalidSetGender	‘setGender’ method should have a String data type argument/parameter. Anything other than a String ‘gender’’ would cause the ‘IllegalArgumentException’ to be flagged.	gender: Male or 345321 or -1 (Non-String = Invalid)

Class #2: ReliefService:

Test Name	Description	Test Value(s)
testReliefServiceConstructor	The ‘ReliefService’ constructor is responsible for initializing the attributes of each new instance of the ReliefService class object with the given input values for the argument/parameters used.	ReliefService(inquirer: Inquirer, missingPerson: DisasterVictim, dateOfInquiry:String, infoProvided: String , lastKnownLocation: Location)
testSetInquirer	Test to check if the ‘setInquirer’ method correctly updates and sets the reference of the ReliefService class to the new instance of the Inquirer class object through the use of the argument/parameter ‘inquirer’ to take the proper given input values.	Inquirer Class Object: Inquirer(“Soummadip”, “Sarkar”, “Information Provided”, “780-1234-567”)
testSetMissingPerson	Test to check if the ‘setMissingPerson’ method correctly updates and sets the reference of the ReliefService	DisasterVictim Class Object: DisasterVictim(“Soummadip”, “2025-04-04”, “2004-06-13”)

	class to the new instance of the DisasterVictim class object through the use of the argument/parameter ‘missingPerson’ to take the proper given input values.	
testSetDateOfInquiry	Test to check if the ‘setDateOfInquiry’ method correctly updates and sets the String data type attribute for the date of inquiry in the ReliefService class attributes through the use of the String data type argument/parameter ‘dateOfInquiry’ to take the proper given input values.	String: “2025-08-08”
testInvalidSetDateOfInquiry	The ‘setDateOfInquiry’ method should only be able to take an argument/parameter of strictly the String data type as it can only update and set the ReliefService class’s attribute to a String data type. Anything other than a String data type should result in the ‘IllegalArgumentException’ to be flagged.	Invalid (Non-String): 2025-08-08
testSetInfoProvided	Test to check if the ‘setInfoProvided’ method correctly updates and sets the String data type attribute for the information provided by the inquirer in the ReliefService class attributes through the use of the String data type argument/parameter ‘infoProvided’ to take the proper given input values.	String: “Male, 5 Feet 8 Inches, Brown, Black Hair, South Asian, Last Name Is Sarkar”
testSetLastKnownLocation	setLastKnownLocation() takes a location object which contains name and address strings of the individual's last known location. The respective getter will be used to check for correct implementation. Should not be possible to have different data types or incorrect name/address, else exception is raised.	Reference To An Instance Of Location Class Object: Location(“University Of Calgary”, “2500 University Dr. NW”)
testGetLogDetails	Test to check if the ‘getLogDetails’ method	“Soummadip entered the University of Calgary shelter

	properly prints or outputs the String data type log details for the ReliefService class	with his Mother, Father, and Siblings”

Class #3: MedicalRecord:

Test Name	Description	Test Value(s)
testMedicalRecordConstructor	The ‘MedicalRecord’ class constructor is responsible for initializing the attributes of each new instance of the MedicalRecord class object with the given input values for the argument/parameters used in the constructor.	MedicalRecord(Location(“Foothills Hospital”, “1403 29 St NW”), “Torn Achilles Tendon, Surgery and Physio Administered”, “2024-08-08”)

testInvalidMedicalRecordConstructor	The ‘MedicalRecord’ class constructor must take the attribute arguments/parameters of data types Location and String, anything else will result in the ‘IllegalArgumentException’ to be flagged.	(Invalid Data Types And Instance Of Location Class Object Type Was Not Used):
		
		MedicalRecord(“Hello World”, 1234, 5678)

testSetLocation	Test to check if the ‘setLocation’ method correctly updates and sets the location attribute reference in the MedicalRecord class through the Location class type argument/parameter variable ‘location’ given the proper input values.	Reference To New Instance Of Location Class Object: Location(“University of Calgary”, 2500 University Dr. NW”)
testInvalidGetTreatmentDetails	The ‘getTreatmentDetails’ method should return the String data type attribute of the MedicalRecord class ‘treatmentDetails’ variable value, if any other data type other than a String value is returned then there would be an ‘IllegalArgumentException’ flagged.	(Non-String Data Type = Invalid Throw Exception): Torn Achilles Tendon, Surgery and Physio Administered 102939
testSetTreatmentDetails	Test to check if the ‘setTreatmentDetails’ method correctly updates and sets the String data type treatment details attribute of the MedicalRecord class through the use of the	String: “Torn Achilles Tendon, Surgery and Physio Administered”

	argument/parameter variable ‘treatmentDetails’ given the proper input values.	
testSetDateOfTreatment	Test to check if the ‘setDateOfTreatment’ method	String: “2024-08-08”
testInvalidSetDateOfTreatment	The ‘setDateOfTreatment’ method should only be able to update and set the ‘dateOfTreatment’ attribute of the MedicalRecord class with a String data type value, if anything other than a String data type is set then this will cause the ‘IllegalArgumentException’ to be flagged.	(Non-String Data Type = Invalid Throw Exception): 2024-04-04

Class #4:Supply:

Test Name	Description	Test Value(s)
testSupplyConstructor	The ‘Supply’ class constructor is responsible for initializing the attributes of each new instance of the Supply class object with the given input values for the argument/parameters used in the constructor.	Supply(“Canned Soup”, 4)
testSetType	Test to check if the ‘setType’ method correctly updates and sets the String data type attribute ‘type’ of the Supply class describing the type of supply and the updates and sets should be able to complete through the String argument/parameter variable ‘type’ accepting String data type input values.	String: “Toilet Paper”
testSetQuantity	Test to check if the ‘setQuantity’ method correctly updates and sets the Integer data type attribute ‘quantity’ of the Supply class describing the quantity of supply and the updates and sets should be	Integer: 16

	able to complete through the Integer argument/parameter variable ‘type’ accepting Integer data type input values.	

Class #5: Location:

Test Name	Description	Test Value(s)
testLocationConstructor	The ‘Location’ class constructor is responsible for initializing the attributes of each new instance of the Location class object with the given input values for the argument/parameters used in the constructor.	Location(“University Of Calgary”, “2500 University Dr. NW”)
testSetName	Test to check if the ‘setName’ method correctly updates and sets the String data type attribute ‘name’ of the Location class describing the name of the evacuation location and the updates and sets should be able to complete through the String argument/parameter variable ‘name’ accepting String data type input values.	String: “University Of Calgary”
testSetAddress	Test to check if the ‘setAddress’ method correctly updates and sets the String data type attribute ‘address’ of the Location class describing the address of the evacuation location and the updates and sets should be able to complete through the String argument/parameter variable ‘address’ accepting String data type input values.	String: “2500 University Dr. NW”
testSetOccupants	Test to check if the ‘setOccupants’ method correctly updates and sets the array of DisasterVictim class objects type attribute ‘occupants’ of the Location class describing the array of instances of the DisasterVictim class objects being currently referenced to	Accepts Reference To New Instances Of Array DisasterVictim Class Objects: [DisasterVictim(“SpongeBob”,” 2025-06-06”, “2000-04-04”), DisasterVictim(“Patrick”, “2025-06-06”)]

	and the updates and sets should be able to complete through the array of DisasterVictim class objects type argument/parameter variable ‘occupants’ accepting existing instances of DisasterVictim class objects as input values.	
testSetSupplies	Test to check if the ‘setSupplies’ method correctly updates and sets the array of Supply class objects type attribute ‘supplies’ of the Location class describing the array of instances of the Supply class objects being currently referenced to and the updates and sets should be able to complete through the array of Supply class objects type argument/parameter variable ‘supplies’ accepting existing instances of Supply class objects as input values.	Accepts Reference To New Instances Of Array Supply Class Objects: [Supply(“Toilet Paper”, 4), Supply(“Canned Vegetables”, 4”)]
testAddOccupant	Within the Location class, the addOccupant() method is designed to accept an occupant that is of type DisasterVictim and appropriately to append this occupant to the array of occupants associated with the current object. Verification of this functionality can be performed by utilising the getOccupants() method. Additionally, it is important to validate the input to ensure that it adheres to the expected type, specifically checking that the input is indeed of the DisasterVictim type.	DisasterVictim Class Object Used So Add The Following DisasterVictim Class Object: DisasterVictim(“Soummadip”, “2025-06-06”, “2004-06-13”)
testRemoveOccupant	Within the Location class, the removeOccupant() method is designed to accept an occupant that is of type DisasterVictim and accurately removes this occupant from the array of occupants associated with the current object. Verification of this	DisasterVictim Class Object Used So Remove The Following DisasterVictim Class Object: DisasterVictim(“Soummadip”, “2025-06-06”, “2004-06-13”)

	functionality can be conducted by utilising the getOccupants() method. Additionally, it is important to validate the input to ensure that it adheres to the expected data type, specifically confirming that the input is indeed of the DisasterVictim data type.	
testAddSupply	Within the Location class, the addSupply method is designed to accept a supply of the data type Supply and appropriately append this supply to the array of supplies associated with the current object. Confirmation of this functionality can be achieved by using the getSupplies() method. Additionally it is essential to validate the input to ensure that it sticks to the expected data type, which should specifically verify that the input is indeed of the Supply data type.	Supply Class Object Used So Add The Following Supply Class Object: Supply(“Water Bottles”, 4)
testRemoveSupply	Within the Location class, the removeSupply method is intended to accept a supply that is of data type Supply and accurately remove this supply from the array of supplies associated with the current object. Verification of this functionality can be performed by utilising the getSupplies() method. Moreover, it is essential to validate the input to ensure that it adheres to the expected type, specifically confirming that the input is indeed of the Supply data type.	Supply Class Object Used So Remove The Following Supply Class Object: Supply(“Toilet Paper”, 4)

Class #6: FamilyRelation:

Test Name	Description	Test Value(s)

testFamilyRelationConstructor	The ‘FamilyRelation’ class constructor is responsible for initializing the attributes of each new instance of the FamilyRelation class object with the given input values for the argument/parameters used in the constructor.	FamilyRelation Constructor Includes Arguments/Parameters Of Other Class Object Type: FamilyRelation(DisasterVictim (“SpongeBob”, “2025-06-04”), “Close Friend”, DisasterVictim(“Patrick”, “2025-06-04”, “2000-04-04”))
testSetPersonOne	Test to check if the ‘setPersonOne’ method correctly updates and sets the DisasterVictim class type attribute ‘personOne’ of the FamilyRelation class describing the instance of the DisasterVictim class object being currently referenced to and the updates and sets should be able to complete through the DisasterVictim class type argument/parameter variable ‘personOne’ accepting existing instances of DisasterVictim class object input values.	DisasterVictim Class Object: 1. DisasterVictim(“SpongeBob”, “2025-06-04”) 2. DisasterVictim(“Patrick”, “2025-06-04”, “2000-04-04”)
testSetRelationshipTo	Test to check if the ‘setRelationshipTo’ method correctly updates and sets the String data type attribute ‘relationshipTo’ of the FamilyRelation class describing the type of relationship the first disaster victim person has to the second disaster victim person and the updates and sets should be able to complete through the String argument/parameter variable ‘relationshipTo’ accepting String data type input values.	String: “Husband” “Wife” “Mother” “Father” “Child” “Sister” “Brother”
testSetPersonTwo	Test to check if the ‘setPersonTwo’ method correctly updates and sets the DisasterVictim class type attribute ‘personTwo’ of the FamilyRelation class describing the instance of the DisasterVictim class object being currently referenced to	DisasterVictim Class Object: 1. DisasterVictim(“SpongeBob”, “2025-06-04”) 2. DisasterVictim(“Patrick”, “2025-06-04”, “2000-04-04”)

	and the updates and sets should be able to complete through the DisasterVictim class type argument/parameter variable ‘personTwo’ accepting existing instances of DisasterVictim class object input values.	

Class #7: Inquirer:

Test Name	Description	Test Value(s)
testInquirerConstructor	Test to check if the Inquirer class constructor correctly initializes the attributes of a new instance of an Inquirer class object through the argument/parameter variables given the correct input values.	Inquirer(“Soummadip”, “Sarkar”,”5 Feet 7 Inches Brown Male”, “780-1234-567”)