Test Decriptions, List of Faults, and Recommended Fixes					
Test Function	**Description**	**Expected Results**	**Faults Found**	**Recommended Fix**	
Main	Precondition: Run CoffeeMaker Enter: Menu option 0, "Exit"	Program Exits			
Main	Precondition: Run CoffeeMaker Enter: Menu option 1, "Add a recipe "	Add Recipe Functionality			
Main	Precondition: Run CoffeeMaker Enter: Menu option 2, "Delete a recipe "	Delete Recipe Functionality			
Main	Precondition: Run CoffeeMaker Enter: Menu option 5, "Edit a recipe "	Edit Recipe Functionality			
Main	Precondition: Run CoffeeMaker Enter: Menu option 4, "Add inventory"	Add Inventory Functionality			
Main	Precondition: Run CoffeeMaker Enter: Menu option 5, "Check inventory"	Inventory Displays			
Main	Precondition: Run CoffeeMaker Enter: Menu option 6, "Make coffee"	Make Coffee Functionality			
setUp	Precondition: Run CoffeeMaker Enter: Menu option 1, "Add a recipe " Name: Coffee Price: 50 Coffee: 3 Milk: 1 Sugar: 1 Chocolate: 0 Return to main menu.	Coffee successfully added.			
setUp	Precondition: Run CoffeeMaker, addRecipe1 Enter: Menu option 1, "Add a recipe " Name: Coffee Price: 50 Coffee: 3 Milk: 1 Sugar: 1 Chocolate: 0 Return to main menu.	Coffee could not be added.			
testAddInvalidRecipes	Precondition: Run CoffeeMaker Enter: Menu option 1, "Add a recipe " Name: Mocha Price: -50 Return to main menu.	Mocha could not be added. Price can not be negative.			
testAddInvalidRecipes	Precondition: Run CoffeeMaker Enter: Menu option 1, "Add a recipe " Name: Mocha Price: 60 Coffee: -3 Return to main menu.	Mocha could not be added. Units of coffee can not be negative.			
testAddInvalidRecipes	Precondition: Run CoffeeMaker Enter: Menu option 1, "Add a recipe " Name: Mocha Price: 60 Coffee: 3 Milk: -2 Return to main menu.	Mocha could not be added. Units of milk can not be negative.			
testAddInvalidRecipes	Precondition: Run CoffeeMaker Enter: Menu option 1, "Add a recipe " Name: Mocha Price: 60 Coffee: 3 Milk: 2 Sugar: -2 Return to main menu.	Mocha could not be added. Units of sugar can not be negative.			

testAddInvalidRecipes	Precondition: Run CoffeeMaker Enter: Menu option 1, "Add a recipe " Name: Mocha Price: 60 Coffee: 3 Milk: 2 Sugar: 2 Chocolate: -3 Return to main menu.	Mocha could not be added. Units of chocolate can not be negative.			
testAddInvalidRecipes	Precondition: Run CoffeeMaker Enter: Menu option 1, "Add a recipe " Name: Mocha Price: a Return to main menu.	Please input an integer.			
testAddInvalidRecipes	Precondition: Run CoffeeMaker Enter: Menu option 1, "Add a recipe " Name: Mocha Price: 60 Coffee: a Return to main menu.	Please input an integer..			
testAddInvalidRecipes	Precondition: Run CoffeeMaker Enter: Menu option 1, "Add a recipe " Name: Mocha Price: 60 Coffee: 3 Milk: a Return to main menu.	Please input an integer.			
testAddInvalidRecipes	Precondition: Run CoffeeMaker Enter: Menu option 1, "Add a recipe " Name: Mocha Price: 60 Coffee: 3 Milk: 2 Sugar: a Return to main menu.	Please input an integer.			
testAddInvalidRecipes	Precondition: Run CoffeeMaker Enter: Menu option 1, "Add a recipe " Name: Mocha Price: 60 Coffee: 3 Milk: 2 Sugar: 2 Chocolate: a Return to main menu.	Please input an integer.			
setUp	Precondition: Run CoffeeMaker, addRecipe1 Enter: Menu option 1, "Add a recipe " Name: Mocha Price: 60 Coffee: 3 Milk: 2 Sugar: 2 Chocolate: 3 Return to main menu.	Coffee successfully added.			
setUp	Precondition: Run CoffeeMaker, addRecipe13 Enter: Menu option 1, "Add a recipe " Name: Latte Price: 60 Coffee: 3 Milk: 3 Sugar: 2 Chocolate: 0 Return to main menu.	Coffee successfully added.			

setUp	Precondition: Run CoffeeMaker, addRecipe14 Enter: Menu option 1, "Add a recipe " Name: Hot Chocolate Price: 60 Coffee: 0 Milk: 2 Sugar: 2 Chocolate: 3 Return to main menu.	Coffee could not be added.			

testAddAndDeleteRecipes	Precondition: addRecipe1 has run successfully
	Enter: Menu option 2, "Delete a recipe "
	Select: Coffee
	Return to main menu.
		Successfully deleted	deleteRecipe() in RecipeBook.java was not properly deleting the recipe. It was setting the index of the recipe to be deleted equal to a new recipe instead of null	Change:
				recipeArray[recipeToDelete] = new Recipe();
				to
				recipeArray[recipeToDelete] = null;	

testAddAndDeleteRecipes	Precondition: Run CoffeeMaker Enter: Menu option 2, "Delete a recipe " Return to main menu.	There are no recipes to delete			

testEditRecipe	Precondition: addRecipe1 has run successfully
	Enter: Menu option 3, "Edit a recipe "
	Select: Coffee
	Price: 50
	Coffee: 3
	Milk: 1
	Sugar: 1
	Chocolate: 0
	Return to main menu.
		Coffee successfully added.	editRecipe() in RecipeBook.java was removing the new recipe's name by setting it equal to an empty string.	Change:
				newRecipe.setName("");
				to
				newRecipe.setName(recipeName);	

testEditRecipeNull	Precondition: addRecipe1 has run successfully Enter: Menu option 3, "Edit a recipe " Select: Coffee Price: 50 Coffee: 3 Milk: 1 Sugar: 1 Chocolate: 0 Return to main menu.	Coffee could not be edited.			
testAddInvalidRecipes	Precondition: addRecipe1 has run successfully Enter: Menu option 3, "Edit a recipe " Select: Coffee Price: -50 Return to main menu.	Coffee could not be edited. Price can not be negative.			editRecipe3-12 cannot even happen because an exception is thrown when trying to create a recipe with invalid values.
testAddInvalidRecipes	Precondition: addRecipe1 has run successfully Enter: Menu option 3, "Edit a recipe " Select: Coffee Price: 60 Coffee: -3 Return to main menu.	Coffee could not be edited. Units of coffee can not be negative.			
testAddInvalidRecipes	Precondition: addRecipe1 has run successfully Enter: Menu option 3, "Edit a recipe " Select: Coffee Price: 60 Coffee: 3 Milk: -2 Return to main menu.	Coffee could not be edited. Units of milk can not be negative.			
testAddInvalidRecipes	Precondition: addRecipe1 has run successfully Enter: Menu option 3, "Edit a recipe " Select: Coffee Price: 60 Coffee: 3 Milk: 2 Sugar: -2 Return to main menu.	Coffee could not be edited. Units of sugar can not be negative.			

testAddInvalidRecipes	Precondition: addRecipe1 has run successfully Enter: Menu option 3, "Edit a recipe " Select: Coffee Price: 60 Coffee: 3 Milk: 2 Sugar: 2 Chocolate: -3 Return to main menu.	Coffee could not be edited. Units of chocolate can not be negative.			
testAddInvalidRecipes	Precondition: addRecipe1 has run successfully Enter: Menu option 3, "Edit a recipe " Select: Coffee Price: a Return to main menu.	Please input an integer.			
testAddInvalidRecipes	Precondition: Run CoffeeMaker Enter: Menu option 3, "Edit a recipe " Select: Coffee Price: 60 Coffee: a Return to main menu.	Please input an integer.			
testAddInvalidRecipes	Precondition: Run CoffeeMaker Enter: Menu option 3, "Edit a recipe " Select: Coffee Price: 60 Coffee: 3 Milk: a Return to main menu.	Please input an integer.			
testAddInvalidRecipes	Precondition: addRecipe1 has run successfully Enter: Menu option 3, "Edit a recipe " Select: Coffee Price: 60 Coffee: 3 Milk: 2 Sugar: a Return to main menu.	Please input an integer.			
testAddInvalidRecipes	Precondition: addRecipe1 has run successfully Enter: Menu option 3, "Edit a recipe " Select: Coffee Price: 60 Coffee: 3 Milk: 2 Sugar: 2 Chocolate: a Return to main menu.	Please input an integer.			
testCheckInventory	Precondition: Run CoffeeMaker Enter: Menu option 4, "Add inventory" Coffee: 5 Milk: 3 Sugar: 7 Chocolate: 2 Return to main menu.	Inventory successfully added			
testAddInvalidInventory	Precondition: Run CoffeeMaker Enter: Menu option 4, "Add inventory" Coffee: -1 Return to main menu.	Cannot add inventory. Units of coffee can not be negative			
testAddInvalidInventory	Precondition: Run CoffeeMaker Enter: Menu option 4, "Add inventory" Coffee: 5 Milk: -1 Return to main menu.	Cannot add inventory. Units of milk can not be negative			
testAddInvalidInventory	Precondition: Run CoffeeMaker Enter: Menu option 4, "Add inventory" Coffee: 5 Milk: 3 Sugar: -1 Return to main menu.	Cannot add inventory. Units of sugar can not be negative			

testAddInvalidInventory	Precondition: Run CoffeeMaker Enter: Menu option 4, "Add inventory" Coffee: 5 Milk: 3 Sugar: 7 Chocolate: -1 Return to main menu.	Cannot add inventory. Units of chocolate can not be negative			
testAddInvalidInventory	Precondition: Run CoffeeMaker Enter: Menu option 4, "Add inventory" Coffee: a Return to main menu.	Please input an integer.			
testAddInvalidInventory	Precondition: Run CoffeeMaker Enter: Menu option 4, "Add inventory" Coffee: 5 Milk: a Return to main menu.	Please input an integer.			
testAddInvalidInventory	Precondition: Run CoffeeMaker Enter: Menu option 4, "Add inventory" Coffee: 5 Milk: 3 Sugar: a Return to main menu.	Please input an integer.			
testAddInvalidInventory	Precondition: Run CoffeeMaker Enter: Menu option 4, "Add inventory" Coffee: 5 Milk: 3 Sugar: 7 Chocolate:a Return to main menu.	Please input an integer.			
testCheckInventory	Precondition:Run CoffeeMaker Enter: Menu option 5, "Check inventory" Return to main menu.	Coffee: 15 Milk: 15 Sugar: 15 Chocolate: 15			
testMakeCoffee	Precondition: addRecipe1 has successfully run Enter: Menu option 6, "Make coffee " Select: Coffee Amount: 60 Return to main menu. Enter: Menu option 5, "Check inventory " Return to main menu.	Your change is 10. Coffee: 12 Milk: 14 Sugar: 14 Chocolate: 15			
testMakeCoffeeBad2	Precondition: addRecipe1 has successfully run Enter: Menu option 3, "Make coffee " Select: Coffee Amount: 40 Return to main menu. Enter: Menu option 5, "Check inventory " Return to main menu.	Your change is 40 Coffee: 15 Milk: 15 Sugar: 15 Chocolate: 15			
testMakeCoffeeBad1	Precondition: Run CoffeeMaker Enter: Menu option 1, "Add a recipe." Name: Coffee Price: 50 Coffee: 16 Milk: 1 Sugar: 1 Chocolate: 0 Return to main menu. Enter: Menu option 6, "Make coffee" Price: 50 Return to main menu.	Your change is 50			
TestgetChocolate	Return the amount of chocolate in the inventory. Chocolate: 15	15			
TestsetChocolate	Set the amount of chocolate in the inventory. Set: 10	10			

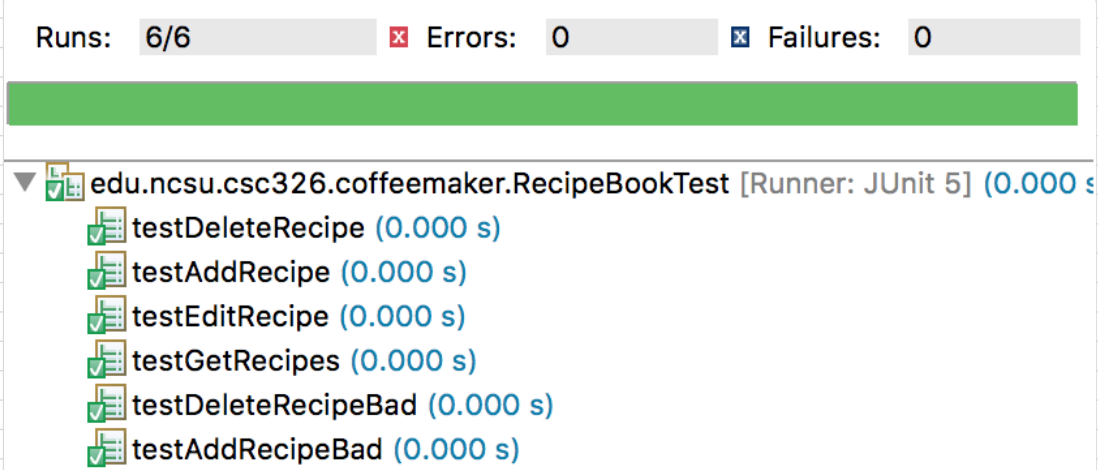
TestsetChocolateBad	Set the amount of chocolate in the inventory. Set: -1 Chocolate: 10	10			
TestaddChocolate	Add chocolate to the inventory. Chocolate: 10 Add: "1"	11			
TestaddChocolateBad	Add chocolate to the inventory. Chocolate: 10 Add: "asdf"	Units of chocolate must be a positive integer			
TestaddChocolateBad1	Add chocolate to the inventory. Chocolate: 10 Add: "-1"	Units of chocolate must be a positive integer			
TestgetCoffee	Return the amount of coffee in the inventory. Coffee: 15	15			
TestsetCoffee	Set the amount of coffee in the inventory. Set: 15	15			
TestsetCoffeeBad	Set the amount of coffee in the inventory. Set: -1 Coffee: 15	15			
TestaddCoffee	Add coffee to the inventory Coffee: 15 Add: "1"	16			
TestaddCoffeeBad2	Add coffee to the inventory. Coffee: 10 Add: "asdf"	Units of coffee must be a positive integer			
TestaddCoffeeBad	Add coffee to the inventory. Coffee: 10 Add: "-1"	Units of coffee must be a positive integer			
TestgetMilk	Return the amount of milk in the inventory. Milk: 15	15			
TestsetMilk	Set the amount of milk in the inventory. Set: 10	10			
TestsetMilkBad	Set the amount of milk in the inventory. Set: -1 Milk: 10	10			
TestaddMilk	Add milk to the inventory. Milk: 10 Add: "1"	11			
TestaddMilkBad	Add milk to the inventory. Milk: 10 Add: "-1"	Units of milk must be a positive integer			
TestaddMilkBad2	Add milk to the inventory. Milk: 10 Add: "wqfaf"	Units of milk must be a positive integer			
TestgetSugar	Return the amount of sugar in the inventory. Sugar: 15	15			
TestsetSugar	Set the amount of sugar in the inventory. Set: 10	10			
TestsetSugarBad	Set the amount of sugar in the inventory. Set: -1	10			

TestaddSugar	Add sugar to the inventory.
	Sugar: 10
	Add: "1"
		11	addSugar() in Inventory.java was not properly adding sugar. The function was checking to see if the amount being added was less than or equal to 0. In essence, this function would only subtract from the total amount of sugar, not add to it	Change
				if (amtSugar <= 0) {
				to
				if (amtSugar >= 0) {	

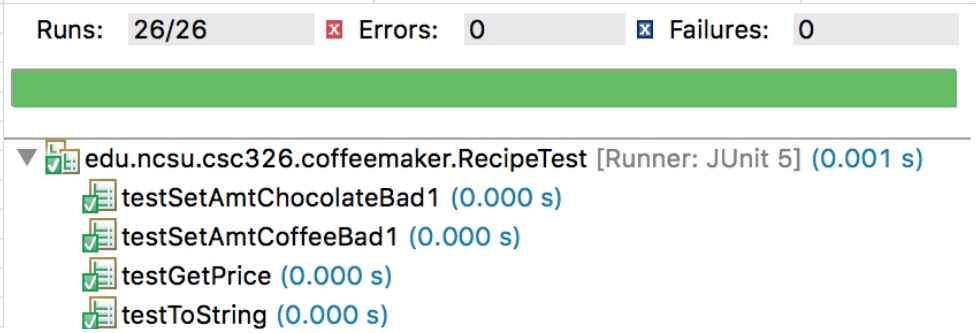
TestaddSugarBad	Add sugar to the inventory. Sugar: 10 Add: "-1"	Units of sugar must be a positive integer			
TestaddSugarBad2	Add sugar to the inventory. Sugar: 10 Add: "fws"	Units of sugar must be a positive integer			
TestenoughIngredients	Checks if there are enough ingredients for a recipe Coffee: 3 Milk: 1 Sugar: 4 Chocolate: 2	TRUE			
TestenoughIngredients	Checks if there are enough ingredients for a recipe Coffee: 10000 Milk: 10000 Sugar: 10000 Chocolate: 10000	FALSE			
TestenoughIngredientsBad	Checks if there are enough ingredients for a recipe Coffee: 0 Milk: 1 Sugar: -3 Chocolate: -4	Units of sugar must be a positive integer			
TestenoughIngredientsBad1	Checks if there are enough ingredients for a recipe Coffee: 18 Milk: 1 Sugar: 1 Chocolate: 1	FALSE			
TestenoughIngredientsBad2	Checks if there are enough ingredients for a recipe Coffee: 1 Milk: 18 Sugar: 1 Chocolate: 1	FALSE			
TestenoughIngredientsBad3	Checks if there are enough ingredients for a recipe Coffee: 1 Milk: 1 Sugar: 18 Chocolate: 1	FALSE			
TestenoughIngredientsBad4	Checks if there are enough ingredients for a recipe Coffee: 1 Milk: 1 Sugar: 1 Chocolate: 18	FALSE			
testUseIngredients	Test that when a recipe is made, the ingredients from the recipe are subtracted from the total ingredients in the inventory.	The total number of each ingredient in the inventory should decrease by the amount of the ingredients in the recipe being made.	useIngredients() in Inventory.java was adding to the toal amount of coffee instead of subtracting from the total amount.	Change Inventory.coffee += r.getAmtCoffee(); to Inventory.coffee -= r.getAmtCoffee();	

testGetRecipes	Returns the recipes in the recipe book. Add recipe r1 Add recipe r2	[r1, r2]
testAddRecipe	Add recipe to the recipe book. Add recipe r1 Add recipe r2 Add recipe r3	TRUE
testAddRecipeBad	Add recipe to the recipe book. Add recipe r1 Add recipe r2 Add recipe r3 Add recipe r1	FALSE
testDeleteRecipe	Remove a recipe from the recipe book. Delete 0 (recipe for coffee)	Coffee
testDeleteRecipeBad	Remove a recipe from the recipe book. Delete 0 (null)	null
testEditRecipe	Edit a recipe at the given index. Edit recipe 1 to be recipe r4 (Mocha)	Mocha

RecipeBookTest.java



RecipeTest.java



testSetAmtChocolateBad (0.000 s)
testGetAmtMilk (0.000 s)
testGetName (0.000 s)
testSetNameBad (0.000 s)
testSetAmtMilkBad (0.001 s)
testGetAmtChocolate (0.000 s)
testSetAmtSugar (0.000 s)
testGetAmtSugar (0.000 s)
testSetPriceBad (0.000 s)
testSetAmtSugarBad1 (0.000 s)
testGetAmtCoffee (0.000 s)
testSetPriceBad1 (0.000 s)
testSetAmtCoffeeBad (0.000 s)
testSetAmtChocolate (0.000 s)
testHashCode (0.000 s)
testEquals (0.000 s)
testSetAmtMilk (0.000 s)
testSetAmtSugarBad (0.000 s)
testSetName (0.000 s)

Coverage Results

CoffeeMaker.java

CoffeeMaker.java	100.0 %	83	0	83
CoffeeMaker	100.0 %	83	0	83
CoffeeMaker()	100.0 %	11	0	11
addInventory(String, String, Stri...	100.0 %	13	0	13
addRecipe(Recipe)	100.0 %	4	0	4
checkInventory()	100.0 %	3	0	3
deleteRecipe(int)	100.0 %	4	0	4
editRecipe(int, Recipe)	100.0 %	5	0	5
getRecipes()	100.0 %	3	0	3
makeCoffee(int, int)	100.0 %	40	0	40

Inventory.java

	Inventory.java	100.0 %	258	0	258
	Inventory	100.0 %	258	0	258
	Inventory()	100.0 %	15	0	15
	addChocolate(String)	100.0 %	25	0	25
	addCoffee(String)	100.0 %	25	0	25
	addMilk(String)	100.0 %	25	0	25
	addSugar(String)	100.0 %	25	0	25
	enoughIngredients(Recipe)	100.0 %	28	0	28
	getChocolate()	100.0 %	2	0	2
	getCoffee()	100.0 %	2	0	2
	getMilk()	100.0 %	2	0	2
	getSugar()	100.0 %	2	0	2
	setChocolate(int)	100.0 %	5	0	5
	setCoffee(int)	100.0 %	5	0	5
	setMilk(int)	100.0 %	5	0	5
	setSugar(int)	100.0 %	5	0	5
	toString()	100.0 %	59	0	59
	useIngredients(Recipe)	100.0 %	28	0	28
RecipeBook.java					
	RecipeBook.java	100.0 %	102	0	102
	RecipeBook	100.0 %	102	0	102
	RecipeBook()	100.0 %	10	0	10
	addRecipe(Recipe)	100.0 %	49	0	49
	deleteRecipe(int)	100.0 %	20	0	20
	editRecipe(int, Recipe)	100.0 %	20	0	20
	getRecipes()	100.0 %	3	0	3
Recipe.java					
	Recipe.java	96.0 %	215	9	224
	Recipe	96.0 %	215	9	224
	equals(Object)	81.1 %	30	7	37
	hashCode()	89.5 %	17	2	19
	Recipe()	100.0 %	21	0	21
	getAmtChocolate()	100.0 %	3	0	3
	getAmtCoffee()	100.0 %	3	0	3

getAmtMilk()	100.0 %	3	0	3
getAmtSugar()	100.0 %	3	0	3
getName()	100.0 %	3	0	3
getPrice()	100.0 %	3	0	3
setAmtChocolate(String)	100.0 %	24	0	24
setAmtCoffee(String)	100.0 %	24	0	24
setAmtMilk(String)	100.0 %	24	0	24
setAmtSugar(String)	100.0 %	24	0	24
setName(String)	100.0 %	6	0	6
setPrice(String)	100.0 %	24	0	24
toString()	100.0 %	3	0	3

Recipe.java is missing some code coverage in haseCode() and equals() because those functions have sections of code that is only executed if a recipe's name is null. However, the setName() method in Recipe.java has a condition that the name being set cannot be null. Thus some portions of code that only execute when the name is null should never be able to execute in practice so we have less than 100% coverage for Recipe.java.