**Acceptance Tests**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Description** | **Expected Results** | **Actual Results** |
| checkOptions1 | Precondition: Run Coffee Maker  Enter: Menu option 0, “Exit” | System exits. |  |
| checkOptions2 | Precondition: Run Coffee Maker  Enter: Menu option 0, “Add a recipe” | System sets up add a recipe method |  |
| checkOptions3 | Precondition: Run Coffee Maker  Enter: Menu option 0, “Delete a recipe” | System sets up delete a recipe method |  |
| checkOptions4 | Precondition: Run Coffee Maker  Enter: Menu option 0, “Edit a recipe” | System sets up edit a recipe method |  |
| checkOptions5 | Precondition: Run Coffee Maker  Enter: Menu option 0, “Add inventory” | System sets up add inventory method |  |
| checkOptions6 | Precondition: Run Coffee Maker  Enter: Menu option 0, “Check inventory” | System sets up check inventory method |  |
| checkOptions7 | Precondition: Run Coffee Maker  Enter: Menu option 0, “Make coffee” | System sets up make coffee method |  |
| addRecipe1 | Precondition: Run CoffeeMaker  Enter: Menu option 1, "Add a recipe " Name: Coffee Price: 50 Coffee: 6 Milk: 1 Sugar: 1 Chocolate: 0 Return to main menu. | Coffee successfully added. | Recipe added was tested using assert statements:  assertTrue(cm.addRecipe(r1)); |
| addRecipe2 | Precondition: Run CoffeeMaker  Enter: Menu option 1, "Add a recipe " Name: Coffee Price: 50 Coffee: 6 Milk: 1 Sugar: 1 Chocolate: 0 Return to main menu. | Coffee can not be added. | Recipe added was tested using assert statements:  cm.addRecipe(r1);  assertFalse(cm.addRecipe(r2)); |
| addRecipe3 | Precondition: Run CoffeeMaker  Enter: Menu option 1, "Add a recipe " Name: Venti Price: -50 Coffee: -6 Milk: -1 Sugar: -1 Chocolate: -2 Return to main menu. | Coffee can not be added. Price can not be negative. If it is, its value will be 0. | Price was tested using assert statements:  assertEquals(0, r3.getPrice()); |
| addRecipe4 | Precondition: Run CoffeeMaker  Enter: Menu option 1, "Add a recipe " Name: Coffee Price: -50 Coffee: -6 Milk: -1 Sugar: -1 Chocolate:-2  Return to main menu. | Units of coffee must be positive. If any number is negative, a status message is printed, and the user is returned to the main menu. | Units of coffee added were tested using assert statements:  assertEquals(0, r3.getAmtCoffee()); |
| addRecipe5 | Precondition: Run CoffeeMaker  Enter: Menu option 1, "Add a recipe " Name: Coffee Price: -50 Coffee: -6 Milk: -1 Sugar: -1 Chocolate:-2  Return to main menu. | Units of milk must be positive. If any number is negative, a status message is printed, and the user is returned to the main menu. | Units of milk added were tested using assert statements:  assertEquals(0, r3.getAmtMilk()); |
| addRecipe6 | Precondition: Run CoffeeMaker  Enter: Menu option 1, "Add a recipe " Name: Coffee Price: -50 Coffee: -6 Milk: -1 Sugar: -1 Chocolate:-2  Return to main menu. | Units of sugar must be positive. If any number is negative, a status message is printed, and the user is returned to the main menu. | Units of sugar added were tested using assert statements:  assertEquals(0, r3.getAmtSugar()); |
| addRecipe7 | Precondition: Run CoffeeMaker  Enter: Menu option 1, "Add a recipe " Name: Coffee Price: -50 Coffee: -6 Milk: -1 Sugar: -1 Chocolate:-2  Return to main menu. | Units of chocolate must be positive. If any number is negative, a status message is printed, and the user is returned to the main menu. | Units of chocolate added were tested using assert statements:  assertEquals(0, r3.getAmtChocolate()); |
| addRecipe8 | 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. | After three recipes have been added to the system, a fourth one may not be added. User is returned to the main menu. | Recipes added were tested using assert statement:  cm.addRecipe(r1);  cm.addRecipe(r4);  cm.addRecipe(r3);  assertFalse(cm.addRecipe(r2)); |
| addRecipe9 | Precondition: Run CoffeeMaker  Enter: Menu option 1, "Add a recipe " Name: Hot Coffee Price: 50 Coffee: 6 Milk: 1 Sugar: 1 Chocolate: 3 Return to main menu. | Units of coffee have to be integers. If not, a status message is printed, and the user is returned to the main menu. | Units of coffee added were tested using assert statements:  assertSame(6, Integer.valueOf(r4.getAmtCoffee())); |
| addRecipe10 | Precondition: Run CoffeeMaker  Enter: Menu option 1, "Add a recipe " Name: Hot Coffee Price: 50 Coffee: 6 Milk: 1 Sugar: 1 Chocolate: 3 Return to main menu. | Units of milk have to be integers. If not, a status message is printed, and the user is returned to the main menu. | Units of milk added were tested using assert statements:  assertSame(1, Integer.valueOf(r4.getAmtMilk())); |
| addRecipe11 | Precondition: Run CoffeeMaker  Enter: Menu option 1, "Add a recipe " Name: Hot Coffee Price: 50 Coffee: 6 Milk: 1 Sugar: 1 Chocolate: 3 Return to main menu. | Units of sugar have to be integers. If not, a status message is printed, and the user is returned to the main menu. | Units of sugar added were tested using assert statements:  assertSame(1, Integer.valueOf(r4.getAmtSugar())); |
| addRecipe12 | Precondition: Run CoffeeMaker  Enter: Menu option 1, "Add a recipe " Name: Hot Coffee Price: 50 Coffee: 6 Milk: 1 Sugar: 1 Chocolate: 3 Return to main menu. | Units of chocolate have to be integers. If not, a status message is printed, and the user is returned to the main menu. | Units of chocolate added were tested using assert statements:  assertSame(3, Integer.valueOf(r4.getAmtChocolate())); |
| addRecipe13 | Precondition: Run CoffeeMaker  Enter: Menu option 1, "Add a recipe " Name: Coffee Price: 50 Coffee: 6 Milk: 1 Sugar: 1 Chocolate: 3 Return to main menu. | Price must be an integer. | Price was tested using assert statements:  assertSame(50, Integer.valueOf(r4.getPrice())); |
| addRecipe14 | Precondition: Run CoffeeMaker  Enter: Menu option 1, "Add a recipe " Name: Coffee Price: 50 Coffee: 6 Milk: 1 Sugar: 1 Chocolate: 3 Return to main menu. | Show all recipes added. | Recipes added were tested using assert statements:  cm.addRecipe(r1);  cm.addRecipe(r2);  assertNotNull(cm.getRecipes()); |
| addRecipe15 | Precondition: Run CoffeeMaker  Enter: Menu option 1, "Add a recipe " Name: Coffee Price: 50 Coffee: 6 Milk: 1 Sugar: 1 Chocolate: 0 Return to main menu. | Show the recipe’s name. | Recipe name was tested using print statement:  cm.addRecipe(r6);  System.out.println(cm.getRecipeForName(“Dark Roast”));  Note: For some reason assertSame or assertEquals statements were showing errors even though the actual/expected output of these statements showed the answer was right. |
|  |  |  |  |
| deleteRecipe1 | Precondition: addRecipe1 has run successfully Enter: Menu option 2, "Delete a recipe " Select: Coffee Return to main menu. | Successfully deleted | Recipe deleted was tested using assert statements:  cm.addRecipe(r1);  assertTrue(cm.deleteRecipe(r1)); |
| deleteRecipe2 | Enter: Menu option 2, "Delete a recipe " Select: Coffee Return to main menu | Selecting a recipe that has not been added, the user is returned to main menu | Recipe deleted was tested using assert statements:  assertFalse(cm.deleteRecipe(r1)); |
| deleteRecipe3 | Enter: Menu option 2, "Delete a recipe " Select: Coffee Return to main menu | Selecting a null item to delete, user is returned to main menu. | Recipe deleted was tested using assert statements:  assertFalse(cm.deleteRecipe(null)); |
|  |  |  |  |
| editRecipe1 | Precondition: addRecipe1 has run successfully  Enter: Menu option 3, "Edit a recipe " Select: Coffee  Price: 50 Coffee: 6 Milk: 1 Sugar: 1 Chocolate: 0  Return to main menu. | Recipe successfully edited. | Recipe edited was tested using assert statements:  cm.addRecipe(r1);  newRecipe = r1;  newRecipe.setAmtSugar(2);  assertTrue(cm.editRecipe(r1, newRecipe)); |
| editRecipe2 | Enter: Menu option 3, "Edit a recipe " Select: Coffee  Price: 50 Coffee: 3 Milk: 1 Sugar: 1 Chocolate: 0  Return to main menu. | Recipe can not be edited if not added, user is returned to main menu. | Recipe edited was tested using assert statements:  newRecipe = r4;  newRecipe.setAmtSugar(2);  assertFalse(cm.editRecipe(r4, newRecipe)); |
| editRecipe3 | Precondition: addRecipe1 has run successfully  Enter: Menu option 3, "Edit a recipe " Select: Coffee  Price: 50 Coffee: 6 Milk: 1 Sugar: 1 Chocolate: 0  Return to main menu. | Price must be positive. If any number is negative, a status message is printed, and the user is returned to the main menu. | Recipe edited was tested using assert statements:  cm.addRecipe(r2);  newRecipe = r2;  newRecipe.setPrice(10);  assertTrue(cm.editRecipe(r2, newRecipe)); |
| editRecipe4 | Precondition: addRecipe1 has run successfully  Enter: Menu option 3, "Edit a recipe " Select: Coffee  Price: 50 Coffee: 6 Milk: 1 Sugar: 1 Chocolate: 0  Return to main menu. | Price must be positive. If any number is negative, a status message is printed, and the user is returned to the main menu. | Recipe edited was tested using assert statements:  cm.addRecipe(r2);  newRecipe = r2;  newRecipe.setPrice(-20);  cm.editRecipe(r2, newRecipe);  assertEquals(0, newRecipe.getPrice()); |
| editRecipe5 | Precondition: addRecipe1 has run successfully  Enter: Menu option 3, "Edit a recipe " Select: Coffee  Price: -50 Coffee: -6 Milk: -1 Sugar: -1 Chocolate:-2 Return to main menu. | Units of coffee must be positive. If not, a status message is printed and the user is returned to main menu.   |  | | --- | |  | | Units of coffee added were tested using assert statements:  cm.addRecipe(r3);  newRecipe = r3;  cm.editRecipe(r3, newRecipe);  assertEquals(0, r3.getAmtCoffee()); |
| editRecipe6 | Precondition: addRecipe1 has run successfully  Enter: Menu option 3, "Edit a recipe " Select: Coffee  Price: -50 Coffee: -6 Milk: -1 Sugar: -1 Chocolate:-2  Return to main menu. | Units of milk must be positive. If not, a status message is printed and the user is returned to main menu. | Units of milk added were tested using assert statements:  cm.addRecipe(r3);  newRecipe = r3;  cm.editRecipe(r3, newRecipe);  assertEquals(0, r3.getAmtMilk()); |
| editRecipe7 | Precondition: addRecipe1 has run successfully  Enter: Menu option 3, "Edit a recipe " Select: Coffee  Price: -50 Coffee: -6 Milk: -1 Sugar: -1 Chocolate:-2  Return to main menu. | Units of sugar must be positive. If not, a status message is printed and the user is returned to main menu. | Units of sugar added were tested using assert statements:  cm.addRecipe(r3);  newRecipe = r3;  cm.editRecipe(r3, newRecipe);  assertEquals(0, r3.getAmtSugar()); |
| editRecipe8 | Precondition: addRecipe1 has run successfully  Enter: Menu option 3, "Edit a recipe " Select: Coffee  Price: -50 Coffee: -6 Milk: -1 Sugar: -1 Chocolate: -2 Return to main menu. | Units of chocolate must be positive. If not, a status message is printed and the user is returned to main menu. | Units of chocolate added were tested using assert statements:  cm.addRecipe(r3);  newRecipe = r3;  cm.editRecipe(r3, newRecipe);  assertEquals(0, r3.getAmtChocolate()); |
|  |  |  |  |
| addInventory1 | Precondition: Run CoffeeMaker Enter: Menu option 4, "Add Inventory "  Coffee: 6 Milk: 7 Sugar: 8 Chocolate: 9 Return to main menu. | Inventory successfully added. | Inventory added was tested using assert statements:  cm.addInventory(6,7,8,9);  assertNotNull(i); |
| addInventory2 | Precondition: Run CoffeeMaker Enter: Menu option 4, "Add Inventory "  Coffee: 6 Milk: 7 Sugar: 8 Chocolate: 9 Return to main menu. | Inventory successfully added | Inventory added was tested using assert statements:  assertTrue(cm.addInventory(6,7,8,9)); |
| addInventory3 | Precondition: Run CoffeeMaker Enter: Menu option 4, "Add Inventory "  Coffee: -6 Milk: 7 Sugar: 8 Chocolate: 9 Return to main menu. | Units of coffee must be positive. If not, a status message is printed and the user is returned to main menu | Units of coffee added were tested using assert statements:  assertFalse(cm.addInventory (-6,7,8,9));  assertEquals(0, i.getCoffee());  assertEquals(7, i.getMilk()); |
| addInventory4 | Precondition: Run CoffeeMaker Enter: Menu option 4, "Add Inventory "  Coffee: 6 Milk: -7 Sugar: 8 Chocolate: 9 Return to main menu. | Units of milk must be positive. If not, a status message is printed and the user is returned to main menu | Units of milk added were tested using assert statements:  assertFalse(cm.addInventory (6,-7,8,9));  assertEquals(0, i.getMilk());  assertEquals(8, i.getSugar()); |
| addInventory5 | Precondition: Run CoffeeMaker Enter: Menu option 4, "Add Inventory "  Coffee: 6 Milk: 7 Sugar: -8 Chocolate: 9 Return to main menu. | Units of sugar must be positive. If not, a status message is printed and the user is returned to main menu | Units of sugar added were tested using assert statements:  assertFalse(cm.addInventory (6,7,-8,9));  assertEquals(0, i.getSugar());  assertEquals(9, i.getChocolate()); |
| addInventory6 | Precondition: Run CoffeeMaker Enter: Menu option 4, "Add Inventory "  Coffee: 6 Milk: 7 Sugar: 8 Chocolate: -9 Return to main menu. | Units of chocolate must be positive. If not, a status message is printed and the user is returned to main menu | Units of coffee added were tested using assert statements:  assertFalse(cm.addInventory (6,7,8,-9));  assertEquals(0, i.getChocolate());  assertEquals(6, i.getCoffee()); |
| addInventory7 | Precondition: Run CoffeeMaker Enter: Menu option 4, "Add Inventory "  Coffee: 0 Milk: 0 Sugar: 0 Chocolate: 0 Return to main menu. | Inventory can be empty. | Zero values added were tested using assert statements:  assertTrue(cm.addInventory (0,0,0,0)); |
|  |  |  |  |
| checkInventory1 | Precondition:  Run CoffeMaker Enter: Menu option 5, "Check Inventory "  Return to main menu | Listing of ingredients inventory shown to user. | Inventory checking was tested using assert statements:  assertNotNull(i); |
| checkInventory2 | Precondition:  Run CoffeMaker Enter: Menu option 5, "Check Inventory " Coffee: -12 Milk: -12 Sugar: -12 Chocolate: -12 Return to main menu. | Listing of ingredients inventory shown to user. | Inventory checking was tested using assert statements:  cm.addInventory(-12,-12,-12,-12);  System.out.println(cm.checkInventory()); |
|  |  |  |  |
| purchaseBeverage1 | Precondition: addRecipe1 has run successfully  Enter: Menu option 6, "Make Coffee " Select: Coffee  Amount: 60  Price: 50  Coffee: 6 Milk: 1 Sugar: 1 Chocolate: 0 Return to main menu | Check to see enough money was deposited to make a purchase. If yes, the beverage will be dispensed. If not, the money will be returned and the user will be returned to main menu. | Purchase Beverage was tested using assert statements:  cm.addInventory(6,7,8,9);  assertEquals(10, cm.makeCoffee(r1, 60); |
| purchaseBeverage2 | Precondition: addRecipe1 has run successfully  Enter: Menu option 6, "Make Coffee " Select: Coffee  Amount: 40 Price: 50 Coffee: 6 Milk: 1 Sugar: 1 Chocolate: 0 Return to main menu | Check to see enough money was deposited. If not, the money will be returned and the user will be returned to main menu. | Purchase Beverage was tested using assert statements:  cm.addInventory(6,7,8,9);  assertEquals(40, cm.makeCoffee(r1, 40); |
| purchaseBeverage3 | Precondition: addRecipe1 has run successfully  Enter: Menu option 6, "Make Coffee " Select: Coffee  Amount: 0 Price: 50 Coffee: 6 Milk: 1 Sugar: 1 Chocolate: 0 Return to main menu | Check to see enough money was deposited. If not, the money will be returned and the user will be returned to main menu. | Purchase Beverage was tested using assert statements:  cm.addInventory(6,7,8,9);  assertEquals(0, cm.makeCoffee(r1, 50); |
| purchaseBeverage4 | Precondition: addRecipe1 has run successfully  Enter: Menu option 6, "Make Coffee " Select: Coffee  Amount:60 Price: 50 Coffee: 6 Milk: 1 Sugar: 1 Chocolate: 3  Return to main menu | Check to see there are enough ingredients in the inventory to make the selected drink. If not, a message will be displayed, the user’s money will be returned and the user will be returned to main menu. | Purchase Beverage was tested using assert statements:  cm.addInventory(5,0,0,2);  assertEquals(60, cm.makeCoffee(r4, 60); |
| purchaseBeverage5 | Precondition: addRecipe1 has run successfully  Enter: Menu option 6, "Make Coffee " Select: Coffee  Price: 50 Coffee: 6 Milk: 1 Sugar: 1 Chocolate: 0 Return to main menu | Check to see there are enough ingredients in the inventory to make the selected drink. If not, a message will be displayed, the user’s money will be returned and the user will be returned to main menu. | Purchase Beverage was tested using assert statements:  cm.addInventory(6,1,1,0);  assertEquals(10, cm.makeCoffee(r5, 60); |
|  |  |  |  |