Kshama Girish
Case ID: kmg134

CustomDate Class Test

The CustomDate class will be tested as follows: First, three separate CustomDate objects are created, at 3/14/15, 1/31/18, and 12/31/17, each for testing a specific case. First, the result of the accessor methods of date 1 is printed out to show that those methods are working as intended. Then the day is incremented for each date. 3/14/15 should turn over to the next day3/15/15, 1/31/18 should turn over to the next month 2/1/18, and 12/31/17 should turn over to the next year 1/1/18. The results are then printed out using the toString method to show that it is working as intended. Another date object is then created at 3/15/15. Date 1 is then compared to Date 4 and should return true as they are the same. Date 2 and 3 are also compared and should return false as they are not the same. The result is printed out

Test Code:

```
		CustomDate c1 = new CustomDate(14, 3, 15);
		CustomDate c2 = new CustomDate(31, 1, 18);
		CustomDate c3 = new CustomDate(31, 12, 17);

		System.out.println("Result From Accessor Methods Of Date 1: " + c1.getMonth()
+ "/" + c1.getDay() + "/" + c1.getYear());

		c1.incrementDay();
		c2.incrementDay();
		c3.incrementDay();

		System.out.println("Result From Day Incrementation Of Date 1: " + c1.getMonth()
+ "/" + c1.getDay() + "/" + c1.getYear());
		System.out.println("Result From Day Incrementation Of Date 2: " + c2.getMonth()
+ "/" + c2.getDay() + "/" + c2.getYear());
		System.out.println("Result From Day Incrementation Of Date 3: " + c3.getMonth()
+ "/" + c3.getDay() + "/" + c3.getYear());

		System.out.println("Result From  toString Method Of Date 1: " + c1.toString());

		CustomDate c4 = new CustomDate(15, 3, 15);

		System.out.println("Result From Date Equality Function For Date 1 and 4: " +
c1.equals(c4));
```

```
        System.out.println("Result From Date Equality Function For Date 2 and 3: " +
c2.equals(c3));

        System.out.println(" ");
```

Output:

```
Result From Accessor Methods Of Date 1: 3/14/15
Result From Day Incrementation Of Date 1: 3/15/15
Result From Day Incrementation Of Date 2: 2/1/18
Result From Day Incrementation Of Date 3: 1/1/18
Result From  toString Method Of Date 1: 3/15/15
Result From Date Equality Function For Date 1 and 4: true
Result From Date Equality Function For Date 2 and 3: false
```

Stock Class Test

The Stock class will be tested as follows: Two different stock classes will be created to test each constructor to ensure functionality. Then each mutator method will be tested by giving them different values. The accessor methods will then be tested by printing out the values of the stock, and the values altered by the mutator methods should be the same as in the printout. Then the buy method will be tested by purchasing 5 stock at $31 each for a $20 commission. The resulting cost will then be printed out which should be 175. Then the second stock is used, where 10 shares are bought. Then the sell shares option will be tested by selling more shares than the user has. The print out value should be 0. Then shares will be sold again, this time selling 2 shares with a commission of 10. The resulting profit should be 52. There are also 3 shares left. The split method will then be tested, where the first stock will be split by ⅓, resulting in a whole number of 1 share left, and no need to sell stock. The second stock will then be split by ⅓ and the resulting shares are 3.3333333. The fractional share is then sold off at $62 per whole share, resulting in a printout of 20.6666666. Then the dividend method is tested, where the dividend rate of .01 is applied to the 1 share, resulting in a printout of .01. Lastly, the accessor methods are tested in a printout.

Test Code:

```
        Stock s1 = new Stock("*/", 30.5);
        Stock s2 = new Stock("@$", 61, c1, .01);

        s1.setCurrentPrice(31);
        s2.setCurrentPrice(62);

        s1.setDividendRate(.01);
        s2.setDividendRate(.02);
```

```
		s1.setDividendDate(c2);
		s2.setDividendDate(c3);

		System.out.println("Stock 1 After Mutator Methods: " + "Price " + 
s1.getCurrentPrice() + " Dividend Rate: " + s1.getDividendRate() + " Dividend Date: " + 
s1.getDividendDate().toString());

		double costIncrease = s1.buy(5, 20);
		System.out.println("Result From Buying 5 shares of stock 1 at 20 commission: " + 
costIncrease);

		s2.buy(10, 20);
		double profit = s1.sell(1000, 10);
		System.out.println("Result From Selling more shares of stock than the user has: " 
+ profit);
		profit = s1.sell(2, 10);
		System.out.println("Result From Selling 2 shares of stock 1 at 10 commission: " + 
profit);

		double splitProfit1 = s1.split(1, 3);
		System.out.println("Result From Splitting stock 1 into a whole number: " + 
splitProfit1);
		double splitProfit2 = s2.split(1, 3);
		System.out.println("Result (profit) From Splitting stock 2 into a fractional: " + 
splitProfit2);

		double dividend = s1.payDividend();

		System.out.println("Dividend From Stock 1: " + dividend);

		System.out.println("Result From Accessor Methods Of Stock 1: " + "Symbol: " + 
s1.getTickerSymbol() + " Price: " + s1.getCurrentPrice() + " Dividend Rate: " + 
s1.getDividendRate() + " Dividend Date: " + s1.getDividendDate().toString() + " Shares: " + 
s1.getNumberShares() + " Cost Basis: " + s1.getCostBasis() + " Capital Gains: " + 
s1.getCapitalGains());
```

Output:

```
Stock 1 After Mutator Methods: Price 31.0 Dividend Rate: 0.01 Dividend Date: 2/1/18
Result From Buying 5 shares of stock 1 at 20 commission: 175.0
Result From Selling more shares of stock than the user has: 0.0
Result From Selling 2 shares of stock 1 at 10 commission: 52.0
Result From Splitting stock 1 into a whole number: 0.0
Result (profit) From Splitting stock 2 into a fractional: 20.66666666666665
Dividend From Stock 1: 0.01
Result From Accessor Methods Of Stock 1: Symbol: */ Price: 31.0 Dividend Rate: 0.01 Dividend
Date: 2/1/18 Shares: 1.0 Cost Basis: 174.33333333333334 Capital Gains:
51.333333333333336
```

Cash Class Test

The Cash Class will be tested as follows: A cash object is instantiated with an interest rate of .017, and the balance is set to 2000. 3 days are “processed” to ensure that there is some interest earned, and then the result of accessor methods is printed out showing the balance, interest rate, and interest earned. A month is then “processed” and the same is printed out, but this time the interest earned should be zero and the balance should have gone up by the amount of interest earned that month, .0931. Then the mutator method is tested, altering the interest rate to .02 and is printed out as .02.

Test Code:

```
Cash ca1 = new Cash(.017);

ca1.balance = 2000;

ca1.processDay();
ca1.processDay();
ca1.processDay();

System.out.println("Result From Accessor Methods of Cash 1 Before
Month: " + "Balance: " + ca1.getBalance() + " Interest Rate: " + ca1.getInterestRate() + " Interest
Earned: " + ca1.getInterestEarned());
ca1.processMonth();
System.out.println("Result From Accessor Methods of Cash 1 After
Month: " + "Balance: " + ca1.getBalance() + " Interest Rate: " + ca1.getInterestRate() + " Interest
Earned: " + ca1.getInterestEarned());

ca1.setInterestRate(.02);
System.out.println("Result From The Mutator Method Altering Interest
Rates: " +"Interest Rate: " + ca1.getInterestRate());
```

Output:

```
Result From Accessor Methods of Cash 1 Before Month: Balance: 2000.0 Interest Rate:
0.017 Interest Earned: 0.09315068493150686
Result From Accessor Methods of Cash 1 After Month: Balance: 2000.0931506849315
Interest Rate: 0.017 Interest Earned: 0.0
```

```
Result From The Mutator Method Altering Interest Rates: Interest Rate: 0.02
```

Savings Class Test

The Savings Class is tested as such: A savings object is instantiated with an interest rate of .015. The deposit method is then tested, by depositing 2000 into the account. It is then printed out which should be 2000. This process is repeated for the withdraw method, where 2000 is taken out and then printed out. \

Test Code:

```
Savings sa1 = new Savings(.015);

sa1.deposit(2000);
System.out.println("Result From The Mutator Method Depositing Money
of Savings 1: " +"Balance: " + sa1.getBalance());
sa1.withdraw(2000);
System.out.println("Result From The Mutator Method Withdrawing Money
of Savings 1: " + "Balance: " + sa1.getBalance());
System.out.println(" ");
```

Output:

```
Result From The Mutator Method Depositing Money of Savings 1: Balance: 2000.0
Result From The Mutator Method Withdrawing Money of Savings 1: Balance: 0.0
```

Loan Class Test

The loans class is tested as such: a new loan is created with an interest rate of .02, loan limit of 3000, and a penalty of 200. The loam mutator methods are then testing by setting the limit to 4000 and penalty to 300. The loan balance is set to 4100. A day is then “processed” and everything is printed out, including balance, penalty, and interest earned which should be 4100, 300, and .22 respectively. A month is then processed and the interest earned and the penalty should be added to the initial loan balance resulting in a print out the value of 4400.22. This process is then repeated with a second loan object but one which is not overdrafted.

Test Code:

```
Loan l1 = new Loan(.02, 3000, 200);
l1.setLoanLimit(4000);
l1.setOverdraftPenalty(300);
l1.balance = 4100;
l1.processDay();
```

```
                              System.out.println("Result From Accessor Methods of Loan 1 Before
Month: " + "Balance: " + l1.getBalance() + " Overdraft Penalty: " + l1.getOverdraftPenalty() + "
Interest Earned: " + l1.getInterestEarned());
                              l1.processMonth();
                              System.out.println("Result From Accessor Methods of Loan 1 After
Month: " + "Balance: " + l1.getBalance() + " Overdraft Penalty: " + l1.getOverdraftPenalty() + "
Interest Earned: " + l1.getInterestEarned());

                              Loan l2 = new Loan(.02, 1000, 100);
                              l2.balance = 500;
                              l2.processDay();
                              System.out.println("Result From Accessor Methods of Loan 2 Before
Month: " + "Balance: " + l2.getBalance() + " Overdraft Penalty: " + l2.getOverdraftPenalty() + "
Interest Earned: " + l2.getInterestEarned());
                              l2.processMonth();
                              System.out.println("Result From Accessor Methods of Loan 2 After
Month: " + "Balance: " + l2.getBalance() + " Overdraft Penalty: " + l2.getOverdraftPenalty() + "
Interest Earned: " + l2.getInterestEarned());
```

Output:

```
          Result From Accessor Methods of Loan 1 Before Month: Balance: 4100.0 Overdraft
Penalty: 300.0 Interest Earned: 0.22465753424657534
          Result From Accessor Methods of Loan 1 After Month: Balance: 4400.224657534247
Overdraft Penalty: 300.0 Interest Earned: 0.0
          Result From Accessor Methods of Loan 2 Before Month: Balance: 500.0 Overdraft
Penalty: 100.0 Interest Earned: 0.027397260273972605
          Result From Accessor Methods of Loan 2 After Month: Balance: 500.027397260274
Overdraft Penalty: 100.0 Interest Earned: 0.0
```

Customer Class Test

The Customer Class is tested as such: A savings object, Loan object, and two stock objects are instantiated. A customer object is then instantiated with all the objects before and the first stock object. All of the accessor methods print out the proper values for everything in the customer class. The first name, last name, and commission mutator methods are then called and values altered. The stock object is changed to the second stock with dividend date 1/30/18. The customer date object is then altered to another date object set to 1/29/18. These are once again printed out and they should match the parameters passed to the mutator methods. 300 dollars are then deposited in the customer savings account to test the deposit method. Then the customer attempts to purchase too many shares and the result is printed out including the customer value which at this time is just the savings deposit. The customer then successfully buys 5 shares at $20 each and a commission of $30. The value of the customer then dropped by the commission to 270 which is printed out to the console. The 5 shares are then sold for the same price but a commission of $30, further dropping the customer value to 240 which is printed out to the console. The customer buys 5 shares again and the customer balance is reset to 300 and the day is incremented, in order to test the dividend payment portion of the increment day method. The before and after are printed out, but the increase should be .075 because of 5 shares at .015 results in an increase of .075. The customer savings balance is then reset to -500 and printed out. The day is incremented again to test the transfer of the negative savings balance to the loan. The print out shows before and after which sets the savings balance to 0 and the loan balance to 500. The customer then deposits 500 and increments the day one last time to turn the month over to 2/1/18 to test the month processing of the loan and savings account. The resulting increase should be .02. The customer savings balance is then reset to 550 and the loan balance set to 500. The customer withdraws 500 and pays it towards the loan. The resulting printout shows 50 left in the savings account and 0 loan balance.

Test Code:

```
                    Stock s3 = new Stock("/c/", 20, c1, .015);
                    Stock s4 = new Stock("/a/", 20, new CustomDate(30, 1, 18), .015);
                    Savings sa2 = new Savings(.015);
                    Loan l3 = new Loan(.015, 5000, 500);

                    Customer customer = new Customer("John", "Titor", s3, sa2, l3, c1, 20);

                    System.out.println("Result From Accessor Methods Of Customer: " +
"Name: " + customer.getFirstName() + " " + customer.getLastName() + " Stock Ticker Symbol: "
+ customer.getStock().getTickerSymbol() + " Savings Interest Rate: " +
customer.getSavings().getInterestRate() + " Loan Interest Rate: " +
customer.getLoan().getInterestRate() + " Commission: " + customer.getCommissionAmount() +
" Date: " + customer.getDate().toString());

                    customer.setFirstName("Bob");
                    customer.setLastName("Dylan");
                    customer.setStock(s4);
                    customer.setCommissionAmount(30);
                    customer.setDate(new CustomDate(29, 1, 18));

                    System.out.println("Result From Accessor Methods Of Customer After
Mutator Calls: " + "Name: " + customer.getFirstName() + " " + customer.getLastName() + "
Stock Ticker Symbol: " + customer.getStock().getTickerSymbol() + " Savings Interest Rate: " +
customer.getSavings().getInterestRate() + " Loan Interest Rate: " +
customer.getLoan().getInterestRate() + " Commission: " + customer.getCommissionAmount() +
" Date: " + customer.getDate().toString());

                    customer.deposit(300);
```

```
				boolean sold = customer.buyShares(1000, "/a/");

				System.out.println("Customer Value After Buying Shares But Failing: " + 
customer.currentValue() + " Actually Sold? " + sold);
				sold = customer.buyShares(5, "/a/");
				System.out.println("Customer Value After Buying Shares And 
Succeeding: " + customer.currentValue() + " Actually Sold? " + sold);

				customer.sellShares(5, "/a/");

				System.out.println("Customer Value After Selling Shares: " + 
customer.currentValue());

				sold = customer.buyShares(5, "/a/");
				customer.getSavings().balance = 300;
				System.out.println("Savings Balance Before Dividend Payed: " + 
customer.getSavings().balance);
				customer.incrementDate();
				System.out.println("Savings Balance After Dividend Payed: " + 
customer.getSavings().balance);

				customer.getSavings().balance = -500;
				System.out.println("Savings Balance Before Negative Balance Transfered 
To Loan: " + customer.getSavings().balance);
				customer.incrementDate();
				System.out.println("Savings Balance After Negative Balance Transfered 
To Loan: " + customer.getSavings().balance + " Loan Balance: " + customer.getLoan().balance);

				customer.deposit(500);
				System.out.println("Savings Balance and Loan Balance Before Month 
Changes: " + "Savings: " + customer.getSavings().balance + " Loan: " + 
customer.getLoan().balance);
				customer.incrementDate();
				System.out.println("Savings Balance and Loan Balance After Month 
Changes: " + "Savings: " + customer.getSavings().balance + " Loan: " + 
customer.getLoan().balance);

				customer.getSavings().balance = 550;
				customer.getLoan().balance = 500;

				System.out.println("Savings Balance and Loan Balance Before Paying 
Loan: " + "Savings: " + customer.getSavings().balance + " Loan: " + 
customer.getLoan().balance);
```

```
                              customer.withdraw(500);
                              customer.payLoan(500);
                              System.out.println("Savings Balance and Loan After Before Paying Loan: 
" + "Savings: " + customer.getSavings().balance + " Loan: " + customer.getLoan().balance);
```

Output:

```
      Result From Accessor Methods Of Customer: Name: John Titor Stock Ticker Symbol: /c/ 
Savings Interest Rate: 0.015 Loan Interest Rate: 0.015 Commission: 20.0 Date: 3/15/15
      Result From Accessor Methods Of Customer After Mutator Calls: Name: Bob Dylan 
Stock Ticker Symbol: /a/ Savings Interest Rate: 0.015 Loan Interest Rate: 0.015 Commission: 
30.0 Date: 1/29/18
      Customer Value After Buying Shares But Failing: 300.0 Actually Sold? false
      Customer Value After Buying Shares And Succeeding: 270.0 Actually Sold? true
      Customer Value After Selling Shares: 240.0
      Savings Balance Before Dividend Payed: 300.0
      Savings Balance After Dividend Payed: 300.075
      Savings Balance Before Negative Balance Transfered To Loan: -500.0
      Savings Balance After Negative Balance Transfered To Loan: 0.0 Loan Balance: 500.0
      Savings Balance and Loan Balance Before Month Changes: Savings: 500.0 Loan: 500.0
      Savings Balance and Loan Balance After Month Changes: Savings: 
500.02054794520546 Loan: 500.02054794520546
      Savings Balance and Loan Balance Before Paying Loan: Savings: 550.0 Loan: 500.0
      Savings Balance and Loan After Before Paying Loan: Savings: 50.0 Loan: 0.0
```