Author: Christopher Tam

**Contractor Class:**

1. **The constructor and most of the getter methods of the class are being tested; the constructor method should be able to initialize the contractor name, contractor address, and contractor contact string values of a created Contractor object. These names should then be returned by the appropriate getter methods. In this case, the returned name, address, and contact values should be “Rockwell”, “7890 W. Garfield”, and “Susie Perkins”, respectively.**

> Contractor a = new Contractor("Rockwell","7890 W. Garfield","Susie Perkins")

> a.getName()

"Rockwell"

> a.getAddress()

"7890 W. Garfield"

> a.getContact()

"Susie Perkins"

1. **Getter method “getAmountPaid” is being tested; should return a preset double value from the “total” class field – in this case, the preset value is 10000.0.**

> Contractor a = new Contractor("Rockwell","7890 W. Garfield","Susie Perkins")

> a.getAmountPaid()

10000.0

1. **Setter methods of the class are tested; these methods should set and update the values of the contractor name, contractor address, and contractor contact. In this case, the aforementioned values should be set to “Automation robotics”, “8709 N. Rockwell”, and “Gerald Eazy”, respectively.**

> Contractor a = new Contractor("Rockwell","7890 W. Garfield","Susie Perkins")

> a.setName("Automation robotics")

> a.setAddress("8709 N. Rockwell")

> a.setContact("Gerald Eazy")

> a.getName()

"Automation robotics"

> a.getAddress()

"8709 N. Rockwell"

> a.getName()

"Automation robotics"

1. **Pay method of the class is tested; the method should add an inputted double value to the existing total class field double value and update this sum as the new total field value. For an input of 300.0, the new value should be 10300.0.**

> Contractor a = new Contractor("Rockwell","7890 W. Garfield","Susie Perkins")

> a.pay(300.0)

> a.getAmountPaid()

10300.0

1. **The toString method of the class is tested; the toString method inherited from Object should be overridden, returning a string in the form “name: address” where name is the contractor name and address is the contractor address. In this case, the method should return “Rockwell: 7890 W. Garfield”.**

> Contractor a = new Contractor("Rockwell","7890 W. Garfield","Susie Perkins")

> a.toString()

"Rockwell: 7890 W. Garfield"

**GovernmentContractor Class:**

1. **The constructor of the class is being tested; the method should create a new GovernmentContractor object with the same initialized values as a Contractor object, inheriting the same methods found in the Contractor class constructor to do so. In this case, the returned name, address, and contact values should be “Laser Electric”, “5678 W. Kick”, and “Biggie Smalls”, respectively.**

> GovernmentContractor a = new GovernmentContractor("Laser Electric","5678 W. Kick","Biggie Smalls")

> a.getName()

"Laser Electric"

> a.getAddress()

"5678 W. Kick"

> a.getContact()

"Biggie Smalls"

1. **The rest of getter, setter, pay, and toString methods of the class are tested. However, the only difference is the toString method is overridden and should now return a string in the form “name address: approved government contractor”. In this case, the values set and returned for the name, address, and contact should be “Yay”, “7890 E. Turtle”, and “Ryan Gosling”, respectively, while the toString method should return "Yay 7890 E. Turtle: approved government contractor".**

> GovernmentContractor a = new GovernmentContractor("Laser Electric","5678 W. Kick","Biggie Smalls")

> a.setName("Yay")

> a.setAddress("7890 E. Turtle")

> a.setContact("Ryan Gosling")

> a.getName()

"Yay"

> a.getAddress()

"7890 E. Turtle"

> a.getContact()

"Ryan Gosling"

> a.getAmountPaid()

10000.0

> a.pay(400.0)

> a.getAmountPaid()

10400.0

> a.toString()

"Yay 7890 E. Turtle: approved government contractor"

**Date class:**

1. **The constructor and getter methods of the class are being tested; the constructor method should be able to initialize the day, month, and year integer values of a created Date object. These values should be returned by the getter methods. In this case, the returned day, month, and year values should be 4, 5, and 17, respectively.**

> Date a = new Date(4,5,17)

> a.getDay()

4

> a.getMonth()

5

> a.getYear()

17

1. **The toString method of the class is tested; it should override the toString method inherited from Object and return a string in the form “month/day/year” where day, month, and year are the initialized values of the Date object. In this case, the method should return “5/4/17”.**

> Date a = new Date(4,5,17)

> a.toString()

"5/4/17"

1. **The equals method of the class is tested; it should override the equals method inherited from Object so that the method returns true if a Date object is equal to the input Object if the input Object is a Date object with the same day, month, and year values as the original Date object, and false otherwise. In this case, the equals method should return true for a and b, and false for a and c.**

> Date a = new Date(4,5,17)

> Date b = new Date(4,5,17)

> Contractor c = new Contractor("Hi","Sup","Yo")

> a.equals(b)

true

> a.equals(c)

false

1. **The daysFromJan1 method of the class is tested; it returns the number of days between a Date and January 1st of the same year. In this case, the Date month is January; number of days returned should be 2.**

> Date a = new Date(3,1,17)

> a.daysFromJan1()

2

1. **The daysFromJan1 method of the class is tested; it returns the number of days between a Date and January 1st of the same year. In this case, the Date month is February; number of days returned in this case should be 33.**

> Date a = new Date(3,2,17)

> a.daysFromJan1()

33

1. **The daysFromJan1 method is tested; In this case, the Date month is March; number of days returned in this case should be 61.**

> Date a = new Date(3,3,17)

> a.daysFromJan1()

61

1. **The daysFromJan1 method is tested; In this case, the Date month is April; number of days returned iin this case should be 92.**

> Date a = new Date(3,4,17)

> a.daysFromJan1()

92

1. **The daysFromJan1 method is tested; In this case, the Date month is May; number of days returned in this case should be 122.**

> Date a = new Date(3,5,17)

> a.daysFromJan1()

122

1. **The daysFromJan1 method is tested; In this case, the Date month is June; number of days returned in this case should be 153.**

> Date a = new Date(3,6,17)

> a.daysFromJan1()

153

1. **The daysFromJan1 method is tested; In this case, the Date month is July; number of days returned in this case should be 183.**

> Date a = new Date(3,7,17)

> a.daysFromJan1()

183

1. **The daysFromJan1 method is tested; In this case, the Date month is August; number of days returned in this case should be 214.**

> Date a = new Date(3,8,17)

> a.daysFromJan1()

214

1. **The daysFromJan1 method is tested; In this case, the Date month is September; number of days returned in this case should be 214.**

> Date a = new Date(3,9,17)

> a.daysFromJan1()

245

1. **The daysFromJan1 method is tested; In this case, the Date month is October; number of days returned in this case should be 275.**

> Date a = new Date(3,10,17)

> a.daysFromJan1()

275

1. **The daysFromJan1 method is tested; In this case, the Date month is November; number of days returned in this case should be 306.**

> Date a = new Date(3,11,17)

> a.daysFromJan1()

306

1. **The daysFromJan1 method is tested; In this case, the Date month is December; number of days returned in this case should be 336.**

> Date a = new Date(3,12,17)

> a.daysFromJan1()

336

1. **The difference method of the class is tested; it returns the difference in days between two inputted Dates, and this difference can be either positive or negative. In this case the years of the two dates are equal, days returned in this case should be -100.**

> Date a = new Date(3,1,17)

> Date b = new Date(13,4,17)

> Date.difference(a,b)

-100

1. **The difference method of the class is tested; in this case the year of date 1 is smaller than the year of date 2, and the month of date 1 is smaller than the month of date 2. Days returned in this case should be -466.**

> Date a = new Date(3,1,16)

> Date b = new Date(13,4,17)

> Date.difference(a,b)

-466

1. **The** **difference method of the class is tested; in this case the year of date 1 is smaller than the year of date 2, the month of date 1 is equal to the month of date 2, and the day of date 1 is larger than the day of date 2. Days returned in this case should be -454.**

> Date a = new Date(3,1,16)

> Date b = new Date(1,4,17)

> Date.difference(a,b)

-454

1. **The** **difference method of the class is tested; in this case the year of date 1 is smaller than the year of date 2, the month of date 1 is equal to the month of date 2, and the day of date 1 is larger than the day of date 2. Days returned in this case should be -454.**

> Date a = new Date(3,1,16)

> Date b = new Date(1,4,17)

> Date.difference(a,b)

-454

1. **The** **difference method of the class is tested; in this case the year of date 1 is smaller than the year of date 2, the month of date 1 is equal to the month of date 2, and the day of date 1 is equal to the day of date 2. Days returned in this case should be -365.**

> Date a = new Date(3,1,16)

> Date b = new Date(3,1,17)

> Date.difference(a,b)

-365

1. **The** **difference method of the class is tested; in this case the year of date 1 is smaller than the year of date 2, the month of date 1 is equal to the month of date 2, and the day of date 1 is smaller than the day of date 2. Days returned in this case should be -368.**

> Date a = new Date(3,1,16)

> Date b = new Date(5,1,17)

> Date.difference(a,b)

-368

1. **The** **difference method of the class is tested; in this case the year of date 1 is smaller than the year of date 2, the month of date 1 is greater than the month of date 2. Days returned in this case should be -217.**

> Date a = new Date(3,6,16)

> Date b = new Date(5,1,17)

> Date.difference(a,b)

-217

1. **The** **difference method of the class is tested; in this case the year of date 1 is larger than the year of date 2, and the month of date 1 is greater than the month of date 2. Days returned in this case should be 246.**

> Date b = new Date(3,1,15)

> Date a = new Date(3,5,16)

> Date.difference(a,b)

246

1. **The** **difference method of the class is tested; in this case the year of date 1 is larger than the year of date 2, the month of date 1 is equal to the month of date 2, the day of date 1 is greater than the day of date 2. Days returned in this case should be 368.**

> Date a = new Date(3,5,16)

> Date b = new Date(1,5,15)

> Date.difference(a,b)

368

1. **The** **difference method of the class is tested; in this case the year of date 1 is larger than the year of date 2, the month of date 1 is equal to the month of date 2, the day of date 1 is equal to the day of date 2. Days returned in this case should be 368.**

> Date a = new Date(3,5,16)

> Date b = new Date(3,5,15)

> Date.difference(a,b)

365

1. **The** **difference method of the class is tested; in this case the year of date 1 is larger than the year of date 2, the month of date 1 is equal to the month of date 2, the day of date 1 is less than the day of date 2. Days returned in this case should be 361.**

> Date a = new Date(3,5,16)

> Date b = new Date(8,5,15)

> Date.difference(a,b)

361

1. **The** **difference method of the class is tested; in this case the year of date 1 is larger than the year of date 2, the month of date 1 is less than the month of date 2. Days returned in this case should be 300.**

> Date a = new Date(3,5,16)

> Date b = new Date(8,7,15)

> Date.difference(a,b)

300

**Bid class:**

1. **The constructor and getter methods of the class are being tested; the constructor method should be able to initialize the contract, contractor, and value of the bid, as a Contract object, a Contractor object, and a double, respectively. These values should be returned by the getter methods. In this case the returned Contract, Contractor, and double should be the address of where the Contract is stored, a string containing the name of the contractor and contractor address, and 500.0, respectively.**

> Contractor a = new Contractor("Spike","7658 N. Rockwell","Susie Derkins")

> Date d = new Date(5,6,9)

> Contract b = new Contract("099864",10.0,300.0,5.0,7.0,d)

> Bid l = new Bid(b,a,500.0)

> l.getContract()

Contract@e2ef48

> l.getContractor()

Spike: 7658 N. Rockwell

> l.getValue()

500.0

**Contract class:**

1. **The constructor and some getter methods are being tested; the constructor method should initialize the contract id, minimum value of the contract, maximum value of the contract, the per day bonus rate of the contract, the per day penalty rate of the contract, and the deadline of the contract, as a string, a double, a double, a double, a double, and a Date object of a created Contract object. These values are returned by the appropriate getter methods.**

> Date a = new Date(5,6,15)

> Contract b = new Contract("0986989",10.0,300.0,5.0,7.0,a)

> b.getID()

"0986989"

> b.getMinValue()

10.0

> b.getMaxValue()

300.0

> b.getBonus()

5.0

> b.getPenalty()

7.0

> b.getDeadline()

6/5/15

1. **The setter methods of the class are tested; these methods should set and update the values of the contract minimum value, contract maximum value, per day bonus rate of the contract, per day penalty rate of the contract, and the deadline of the contract. In this case, it should set the aforementioned values to 4.0, 250.0, 3.0, 6.0, and 8/4/20, respectively.**

> Date a = new Date(5,6,15)

> Contract b = new Contract("0986989",10.0,300.0,5.0,7.0,a)

> Date u = new Date(4,8,20)

> b.setMinValue(4.0)

> b.setMaxValue(250.0)

> b.setBonus(3.0)

> b.setPenalty(6.0)

> b.setDeadline(u)

> b.getMinValue()

4.0

> b.getMaxValue()

250.0

> b.getBonus()

3.0

> b.getPenalty()

6.0

> b.getDeadline()

8/4/20

> b.getID()

"0986989"

1. **The equals method of the class is tested;** **it should override the equals method inherited from Object so that the method returns true if a Contract object is equal to the input Object if the input Object is a Contract object with the same contract ID as the original Contract object, and false otherwise. In this case, the equals method should return true for t and b, and false for t and u.**

> Date u = new Date(4,8,20)

> Contract b = new Contract("0986989",10.0,300.0,5.0,7.0,u)

> Date a = new Date(5,6,15)

> Contract t = new Contract("0986989",8.0,350.0,6.0,4.0,a)

> t.equals(b)

true

> t.equals(u)

false

1. **The isAcceptingBids and awardContract methods of the class are tested; if the awardContract method is not called, the isAcceptingBids method should return true. Once the awardContract method is called, the isAcceptingBids method should return false.**

> Date a = new Date(5,6,15)

> Contract t = new Contract("0986989",8.0,350.0,6.0,4.0,a)

> t.isAcceptingBids()

true

> t.awardContract()

> t.isAcceptingBids()

false

1. **Tests the getBestBid and makeBid methods of the class; getBestBid should return null if makeBid method has not been called. If makeBid method is called and returns true, getBestBid should return a bid address; if makeBid method is called and returns false, getBestBid should return null still.**

> Date u = new Date(4,8,20)

> Contract b = new Contract("0986989",10.0,300.0,5.0,7.0,u)

> Contractor y = new Contractor("Spike Industries","708 N. Clydeburn","Zack Hamilton")

> Bid r = new Bid(b,y,1000.0)

> Bid i = new Bid(b,y,250.0)

> Date a = new Date(5,6,15)

> Contract h = new Contract("0986989",5.0,400.0,5.0,7.0,a)

> h.getBestBid()

null

> h.makeBid(r)

false

> h.getBestBid()

null

> h.makeBid(i)

true

> h.getBestBid()

Bid@1d4c4e5

1. **Tests the getBestBid and makeBid methods of the class; If getBestBid is no longer null, it should be replaced by another Bid object *only if* makeBid returns true again. MakeBid will *only return true* *again* if an inputted Bid has a bid value less than the current best bid’s value. Code below continues from that above.**

> Bid o = new Bid(b,y,100.0)

> h.makeBid(o)

true

> h.getBestBid()

Bid@e2669c

> Bid j = new Bid(b,y,150.0)

> h.makeBid(j)

false

> h.getBestBid()

Bid@e2669c

1. **Tests the setComplete method of the class; if the difference between the inputted date to the setComplete method and the contract deadline (where this difference is calculated via the difference method of the Date class) is negative, setComplete pays the contractor of the best bid an amount equal to the value of the best bid plus a bonus IF this amount is greater than zero and less than the maximum value of the contract. In this case, 10000.0 was already paid to the contractor, so the total amount paid to the contractor after setComplete is executed should be 10243.0. In other words, the amount paid to the contractor should be 243.0.**

> Date a = new Date(5,6,15)

> Contract h = new Contract("0986989",5.0,400.0,5.0,7.0,a)

> Date b = new Date(6,7,13)

> Date u = new Date(4,8,20)

> Contract b = new Contract("0986989",10.0,300.0,5.0,7.0,u)

> Contractor y = new Contractor("Spike Industries","708 N. Clydeburn","Zack Hamilton")

> Bid r = new Bid(b,y,250.0)

> h.makeBid(r)

true

> Date t = new Date(4,6,15)

> h.setComplete(t)

> ((h.getBestBid()).getContractor()).getAmountPaid()

10243.0

1. **Tests the setComplete method of the class; if the difference between the inputted date to the setComplete method and the contract deadline (where this difference is calculated via the difference method of the Date class) is positive, setComplete pays the contractor of the best bid an amount equal to the value of the best bid plus a bonus IF this amount is greater than zero and less than the maximum value of the contract. In this case, 10000.0 was already paid to the contractor, so the total amount paid to the contractor after setComplete is executed should be 10047.0. In other words, the amount paid to the contractor should be 47.0, which is (and should be) less than the bid value of 250.0.**

> Date a = new Date(5,6,15)

> Contract h = new Contract("0986989",5.0,400.0,5.0,7.0,a)

> Date b = new Date(6,7,13)

> Date u = new Date(4,8,20)

> Contract b = new Contract("0986989",10.0,300.0,5.0,7.0,u)

> Contractor y = new Contractor("Spike Industries","708 N. Clydeburn","Zack Hamilton")

> Bid r = new Bid(b,y,250.0)

> h.makeBid(r)

true

>Date t = new Date(4,7,15)

> h.setComplete(t)

>((h.getBestBid()).getContractor()).getAmountPaid()

10047.0

1. **Tests the setComplete method of the class; if the difference between the inputted date to the setComplete method and the contract deadline (where this difference is calculated via the difference method of the Date class) is zero, setComplete pays the contractor of the best bid an amount equal to the value of the best bid IF this value is greater than zero and less than the maximum value of the contract. In this case, 10000.0 was already paid to the contractor, so the total amount paid to the contractor after setComplete is executed should be 10250.0. In other words, the amount paid to the contractor should be 250.0, which is exactly the bid value.**

> Date a = new Date(5,6,15)

> Contract h = new Contract("0986989",5.0,400.0,5.0,7.0,a)

> Date b = new Date(6,7,13)

> Date u = new Date(4,8,20)

> Contract b = new Contract("0986989",10.0,300.0,5.0,7.0,u)

> Contractor y = new Contractor("Spike Industries","708 N. Clydeburn","Zack Hamilton")

> Bid r = new Bid(b,y,250.0)

> h.makeBid(r)

true

> h.setComplete(a)

> ((h.getBestBid()).getContractor()).getAmountPaid()

10250.0

**GovernmentContract class:**

1. **The constructor of the class is being tested; the method should create a new GovernmentContract object with the same initialized values as a Contract object, inheriting the same methods found in the Contract class constructor to do so. In this case, the returned contract id, minimum value of the contract, maximum value of the contract, the per day bonus rate of the contract, the per day penalty rate of the contract, and the deadline of the contract should be “567”, 4.0, 100.0, 6.0, 7.0, and 8/4/20, respectively.**

> Date u = new Date(4,8,20)

> GovernmentContract a = new GovernmentContract("567",4.0,100.0,6.0,7.0,u)

> a.getID()

"567"

> a.getMinValue()

4.0

> a.getMaxValue()

100.0

> a.getBonus()

6.0

> a.getPenalty()

7.0

> a.getDeadline()

8/4/20

1. I was not able to complete the overridden version of the makeBid method inherited from the Contract class. However, if I had completed it, I would test the method by inputting a bid that has a government contractor and check if the method returns a true, and input a bid that has a non-government contractor and check if the method returns a false.