**44-542 Object Oriented Programming**

**Lab02: Classes Lab Activity**

**Objective:** Covers the creation and usage of **Concrete class** and **Scanner** class.

**NOTE:**

* Do not hard code any values.
* Do not use any looping statements for this lab.
* Check the sample output to know how the results need to be printed.
* Read every instruction carefully and follow them strictly.
* Do not change the name of the attributes and methods given below.
* Use **this** keyword wherever necessary
* This lab contains two parts (Part 1 & Part 2). Part 1 🡪 Store class and StoreDriver class, which deals with constructors, getters, setters and toString method.

Part 2 🡪 DoctorSalary and DoctorSalaryDriver, which deals getters, setters and with some mathematical calculations.

Make sure you complete both the parts.

* @author notation should contain your full name.
* Generate the java docs for your project.

**Part 1**

1. Create a New Project and name it as **Lastname\_Lab02Classes** where **Lastname** is your last name.
2. Create a new package in the project created and name it as **stores**.
3. Create a new Java class in **stores** package and name it as **Store**.
4. Write statements to declare the following attributes. Do not add any instance variables beyond those shown here. Access specifiers must be private for all the given instance variables.

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Type** | **Attribute Description** |
| **storeName** | **String** | Name of the store (Note: For this project, we always consider the name of the store has exactly three words.  Examples: Reliance Fresh Walmart (or)  Macys Private Ltd.) |
| **storeID** | **int** | Unique identifier of the store |
| **phoneNumber** | **String** | This is a ten-digit phone number with only numerical values. |
| **street** | **String** | Name of the street where the store is located (This input for this variable should also contain the house number along with the street name  Ex: 654 West Addison Street) |
| **city** | **String** | Name of the city where the store is located |
| **state** | **String** | Name of the state where the store is located (Note: The input for this variable should contain the full state name not the state code or abbreviations) |
| **zipCode** | **String** | This is a five-digit zip code of the state where the store is located |

1. Constructors:
   1. Create a no-argument constructor with no body. This constructor sets the values to default values. Ex: For string, it will set to “” and for double, it’s 0.0 etc…
   2. Create a one arg-constructor with storeID. This constructor sets the storeID to the given instance variable and the other attributes with below sample values.

Store Name: **Unknown**

Street of the Store: **NA**

City of the Store: **NA**

State of the Store: **NA**

Zip code of the Store: **zzzzzz**

Phone number of the Store: **dddddddddd**

* 1. Create a constructor with all the given attributes in the same order, as instance variables are created. The parameters are used to set the values of the instance variables.

1. Methods:
   1. Write the getter and setter methods for each of the instance variable declared.
      1. The setter method of store name should remove the leading and trailing spaces and must contain exactly one space in between the words. Example : Store Name = ” Buffalo Wild Wings ”

formatted Store Name = “Buffalo Wild Wings”

Hint: Use replaceAll() method in the String class.

Resources: Use the Java API shown in the class.

* 1. Write a method **getFormattedAddress()** to return the mailing address of the store. The example below illustrates the store address returned:

108 N State Street

Chicago, Illinois-60018

* 1. Write a method getFormattedPhoneNumber**()** to return the phone number in this format (ddd)ddd-dddd where d is the numerical digit.
  2. Write a method **replaceStoreName(String newName).** This method should replace the old store name with the new store name. Return type of this method should be void. (Hint: Use existing setter method to format and set new storeName.)
  3. Write a method **getFirstWordOfStoreName()**to return the first word in the store name.
  4. Write a method **getMiddleWordOfStoreName()**to return the middle word in the store name.
  5. Write a method **getLastWordOfStoreName()**to return the last word in the store name.
  6. Override the **toString()** method which should be used to display the object. Please see the sample output to know the pattern.

(Note: This method should use **getFormattedAddress(), getFormattedPhoneNumber()**for printing the address and phone number of store)

1. Create a new Java Main class in **stores** package and name it as **StoreDriver**.

|  |
| --- |
| **StoreDriver Class Specifications**  **NOTE: Make sure your output matches with the sample output given.**  1. Print the message “Testing the Store Class”.  2. Declare and create a **Store** object as **store1** with parameterized constructor with  values "Timmy Tommy Enterprises", 1000, "3127869900", "311 Jarvis Square",  "Chicago", "Illinois", "60018" respectively.  3. Print the message “Testing the Getter methods or Accessors of Store Class”.  4. Invoke all the getter methods on **store1** and print the result.  5. Print the message “Testing the toString method”.  6. Invoke the **toString()** method on **store1** and print the result.  7. Print the message “Testing the user defined methods”.  8. Invoke the **getFirstWordOfStoreName()** method on **store1** and print the result.  9. Invoke the **getMidddleWordOfStoreName()** method on **store1** and print the  result.  10. Invoke the **getLastWordOfStoreName()** method on **store1** and print the result.  11. Invoke the **replaceStoreName()** method on **store1** to replace the existing store  name with the “JC Penny Store”.  12. Print the message "Printing the store1 object after invoking the replace method”.  13. Invoke the **toString()** method on **store1** and print the result.  14. Declare and create a new **Store** object as **store2** using no-arg constructor.  15. Print the message “Testing the Getter methods or Accessors of Store Class with no-arg  constructor”.  16. Invoke all the getter methods and print the result.  17. Invoke the **setStoreName()** method on **store2** with the value “KC India Mart”.  18. Invoke the **setStoreID()** method on **store2** with the value 1001.  19. Invoke the **setPhoneNumber()** method on **store2** with the value “9136818080”.  20. Invoke the **setStreet()** method on **store2** with the value “8542 w 133rd Street”.  21. Invoke the **setCity()** method on **store2** with the value “Overland Park”.  22. Invoke the **setState()** method on **store2** with the value “Kansas”.  23. Invoke the **setZipcode()** method on **store2** with the value “66213”.  24. Print the message “Testing the store class using toString after invoking the Setter  methods or Mutators”.  25. Invoke the **toString()** method on **store2** and print the result.  26. Declare and create a new Store object as store3 using one argument constructor with  store id as 1391.  27. Print the message “Testing the Getter methods or Accessors of Store Class with one  argument constructor”.  28. Invoke all the getter methods and print the result.  29. Invoke the replaceStoreName() method on store3 with the value “Buffalo Wild Wings”.  30. Invoke the setPhoneNumber() method on store3 with the value “8163878320”.  31. Invoke the setStreet() method on store3 with the value “5403 N Belt Hwy”.  32. Invoke the setCity() method on store3 with the value "St Joseph”.  33. Invoke the setState() method on store3 with the value “Missouri”.  34. Invoke the setZipcode() method on store3 with the value “64506”.  35. Print the message “Testing the store class using toString after invoking the Setter  methods or Mutators”.  36. Invoke the toString() method on store3 and print the result.  37. Print the message “Testing the Scanner class to take input from the console”.  38. Declare and create a Scanner object as scannerObject to take input from the  console.  39. Print the messages given in the sample run accordingly.  Here is the sample run of the program how the input should be given from the console  Enter the store details  Enter the store name:  Dunkin Donuts Limited  Enter the store ID:  1003  Enter the store phone number:  8088397505  Enter the street name of the store:  3270 Ualena St  Enter the city name, state name, zip code of the store in one line without any commas:  Honolulu Hawaii 96819  40. Declare and create Store object store4 with parameterized constructor  with the values you have taken from the console. .  41. Print the message “Testing the toString method without invoking the method”. |

|  |
| --- |
| **Sample Output Window**  \*\*\*\*\*\*\*\*\*\*\*\*\*\*Testing the Store Class \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Testing the Getter methods or Accessors of Store Class  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Store Name: Timmy Tommy Enterprises  Store ID: 1000  Street of the Store: 311 Jarvis Square  City of the Store : Chicago  State of the Store: Illinois  Zip code of the Store: 60018  Phone number of the Store: 3127869900  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Testing the toString method  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Timmy Tommy Enterprises (1000)  311 Jarvis Square  Chicago, Illinois-60018  (312)786-9900  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Testing the user defined methods  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  First word of store name: Timmy  Middle word of the store name: Tommy  Last word of the store name: Enterprises  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Printing the store1 object after invoking the replace method  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  JC Penny Store (1000)  311 Jarvis Square  Chicago, Illinois-60018  (312)786-9900  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Testing the Getter methods or Accessors of Store Class with no-arg constructor  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Store Name: null  Store ID: 0  Street of the Store: null  City of the Store: null  State of the Store: null  Zip code of the Store: null  Phone number of the Store: null  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Testing the store class using toString after invoking the Setter methods or Mutators  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  KC India Mart (1001)  8542 w 133rd Street  Overland Park, Kansas-66213  (913)681-8080  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Testing the Getter methods or Accessors of Store Class with one argument constructor  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Store Name: Unknown  Store ID: 1391  Street of the Store: NA  City of the Store: NA  State of the Store: NA  Zip code of the Store: zzzzzz  Phone number of the Store: dddddddddd  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Testing the store class using toString after invoking the Setter methods or Mutators  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Buffalo Wild Wings (1391)  5403 N Belt Hwy  St Joseph, Missouri-64506  (816)387-8320  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Testing the scanner class to take input from the console  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Enter the store details  Enter the store name:  Dunkin Donuts Limited  Enter the store ID:  1003  Enter the store phone number:  8088397505  Enter the street name of the store:  3270 Ualena St  Enter the city name, state name, zip code of the store in one line without any commas:  Honolulu Hawaii 96819  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Testing the toString method without invoking the method  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Dunkin Donuts Limited (1003)  3270 Ualena St  Honolulu, Hawaii-96819  (808)839-7505 |

**Part 2**

1. Create another package and name it as **doctorSalary** with in the same project.
2. Create a new Java class in **doctorSalary** package and name it as **DoctorSalary**.
3. Write statements to declare the following attributes. Do not add any instance variables beyond those shown here. Access specifiers must be private for all the given instance variables.

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Type** | **Attribute Description** |
| **hourlyRate** | **double** | Hourly pay rate of the doctor |
| **insuranceRate** | **double** | Insurance percentage |
| **taxRate** | **double** | Tax percentage |
| **HOURS** | **int** | Total hours of work per week(constant), initialize with 40 |

1. Constructor:
   1. Create one constructor with parameters. The parameters are used to set the values of the instance variables.

**public DoctorSalary(double hourlyRate, double insuranceRate, double taxRate)**

* 1. Create one no-argument constructor with an empty body.

1. Methods:
   1. Write getter and settermethods for each of the instance variables declared.
   2. Write a method for calculating the monthly salary of the doctor. Name method as **calcMonthlySalary().** It must return a double value.

**Hint:**

* Calculate based on number of weekly hours provided.
* Assume there are 4 weeks in a month.

*If the hourly pay rate of the doctor is $24.5 then the monthly salary should be $3920.0.*

* 1. Write a method for calculating the monthly insurance. Name the method as **calcMonthlyInsurance().** It must return a double value.

**Hint:** Monthly insurance = Monthly salary \*(insurance rate)/100

*If the monthly salary is $3920.0 and insurance is 12.5% then the monthly insurance should be $490.0*

* 1. Write a method for calculating the annual gross salary. Name the method as **calcAnnualGrossSalary(double bonus)**. It must return a double value.

**Hint:** Annual gross salary = bonus + annual salary

*If the bonus is $3875.9 and the annual salary is $47040.0 then the annual gross salary is $50915.9*

* 1. Write a method for calculating the annual net pay. Name the method as **calcAnnualNetPay(double bonus).** It must return a double value.

**Hint:** Annual net pay = Annual gross salary – annual tax – annual insurance

*If the bonus is 3875.9, tax exempted per annum is 9.25%, and the annual insurance is 5880.0 then the annual Net pay should be $* *40326.17925*

* 1. Write a **toString()** method which should be used to display the values for each attribute. This method should return a String. Please see the sample output for knowing the pattern.

1. Create **DoctorSalaryDriver** class in **doctorSalary** package. Now follow the given instructions

|  |
| --- |
| **DoctorSalaryDriver Class specifications** |
| * 1. Declare and initialize a Scanner object to read from the keyboard. Name it as **scan** |
| * 1. Prompt the values for hourlyRate, insuranceRate, taxRate and bonus for a doctor. Read each value using the Scanner object created in the previous step. |
| * 1. Use the input values to create a new DoctorSalary object named as **docSalObj1** |
| * 1. Print **docSalObj1** using **toString()** method in DoctorSalary class. |
| * 1. Print with appropriate labels, the values returned by **calcMonthlySalary(), calcMonthlyInsurance(), calcAnnualGrossSalary(double bonus), calcAnnualNetPay(double bonus)**. |
| * 1. Create a new DoctorSalary object and name it as **docSalObj2** using the no argument constructor. |
| * 1. Print **docSalObj2** using **toString()** method in DoctorSalary class. |
| * 1. Print with appropriate labels, the values returned by **calcMonthlySalary(), calcMonthlyInsurance(), calcAnnualGrossSalary(double bonus), calcAnnualNetPay(double bonus).** |
| * 1. Use the setter methods to set the values for hourlyRate, insurance Rate, taxRate as $42.85, 15.30%, 11.55%. |
| * 1. Prompt the value for new bonus as $6344.66 and read the value using the scanner object created in the **6.a**. Use this bonus in appropriate methods for calculation. |
| * 1. Print **docSalObj2** using **toString()** method in DoctorSalary class. |
| * 1. Print with appropriate labels, the values returned by **calcMonthlySalary(), calcMonthlyInsurance(), calcAnnualGrossSalary(double bonus), calcAnnualNetPay(double bonus).** |

1. Include Javadoc comments for each constructor, and method in **Doctor** and **DoctorSalary** classes in both **doctors** and **doctorSalary** packages using **@author**, **@param**, and **@return** annotations wherever appropriate. Remember that for each method, the first line of the Javadoc comment should be a brief description of the method.
2. Generate documentation for your project by clicking on **Run** from the NetBeans menu bar and then selecting **Generate Javadoc**. The documentation will be placed in a **javadoc** subfolder of the **dist** subfolder inside your project folder. You can view the documentation created by opening **index.html**.

**Sample Output – Part2**: (User input is in Red)

|  |
| --- |
| Testing the DoctorSalary class:  Enter the hourly pay rate of the Doctor: $24.5  Enter the insurance rate of the Doctor in percentage: 12.5  Enter the tax rate of the doctor in percentage: 9.25  Enter the bonus of the doctor: 3875.9  The details of docSalObj1 object are as follows:  Hourly pay rate: $24.5, insurance rate: 12.5%, tax: 9.25%, Hours per week: 40  The monthly salary of the doctor is: $3920.0  The monthly insurance of the doctor is: $490.0  The annual gross salary of the doctor is: $50915.9  The gross annual net pay of the doctor is: $40326.17925  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  The details of docSalObj2 are as follows:  Hourly pay rate: $0.0, insurance rate: 0.0%, tax: 0.0%, Hours per week: 40  The monthly salary of the doctor is: $0.0  The monthly insurance of the doctor is: $0.0  The annual gross salary of the doctor is: $3875.9  The gross annual net pay of the doctor is: $3875.9  Enter the new bonus of the doctor: 6344.66  Hourly pay rate: $42.85, insurance rate: 15.3%, tax: 11.55%, Hours per week: 40  The monthly salary of the doctor is: $6856.0  The monthly insurance of the doctor is: $1048.968  The annual gross salary of the doctor is: $88616.66  The annual net pay of the doctor is: $65793.81977 |

**Submit you solution by following the steps below:**

* Save your files in **NetBeans**.
* Zip your entire Project. (File name should be ***Lastname*\_Lab02Classes.zip** where **Last name** is your last name.)
* Submit the Zip file to the **Lab02Classes** drop box using file upload option.
* Download the Zip file you have submitted.
* Look in the Zip file and verify the class files in the Zip folder are correct. If not resave your project i24n **NetBeans** and resubmit.