**Home Assignment – 6 (Group Assignment)  
Due Date: November 24th (Monday), 11.59  
Total Points: 100**

**You could work in groups of up to three students each. Need to turn in only one copy per group.**

This assignment will cover chapters 8 and 9 in the text.

**Chapter 8:**

20, 21, 30, 34, 35 (2 points each)

**Programming exercise - I: (30 points)**

In this program you will be reusing the *Date* class you created for Lab Assignment – 8 (You could copy the source code into your new project for this assignment). You will first implement a user defined class called *Employee*. An employee object stores information about an employee. Each employee has a first name, last name, birth date and hire date. You will implement the following the Employee class,

* Instance variables *firstName, lastName, birthDate* (Date type), *hireDate* (Date type)
* Variable to keep track of the count of *Employee* objects created. Note there will be only one copy of this variable that is shared by all the objects of this class
* Create appropriate constructors. Note that to initialize *birthDate a*nd *hireDate* the constructors will need parameters of *Date* type
* Create a method *getEmployeeCount( )* that returns the current count of the Employee objects
* Create a *toString( )* and *equals* method ( )
* Include JavaDoc style comments along with pre-conditions, post-conditions for each of the mutator methods. The comments should also include description of the method along with information about any exceptions the method could possibly generate.

Next you would implement a client program *EmployeeTest* to test the Employee class. Do the following the EmployeeTest class.

* Implement the main method
* Call the *getEmployeeCount( )* method and print it’s return value to the output window.
* Create an array of Employee type of length 5.
* Use a for loop to create to prompt the user for necessary information (names, birth dates and hire dates to create 5 *Employee* objects and assign these Employee objects to the array that you create.
* Use a *for loop* to retrieve one employee object at a time and call it’s *toString( )* method and print the return value on the output window.
* Demonstrate the usage of the *equals( )* method by calling the equals( ) method to compare the Employee object in the 2nd index position and the 4th index position.
* Display the current count of Employee objects by calling *getEmployeeCount( )* method
* **Make sure to keep make your output verbose and comment your code.**

**Chapter 9**

45 (4 points)

47, 49 (3 points each)

**Programming Exercise II:**

Design and code a program including the following classes, as well as a client class to test all the methods coded:

A *Passenger* class - use defined class encapsulating a passenger. A passenger has two attributes: a *name*, and a *class of service*, which will be 1 or 2. Provide an appropriate constructor, get and set methods along with equals( ) and toString( ) methods.

A *Train* class, encapsulating a train of passengers. This class requires only one instance variable: *passengerList* of *ArrayList<Passenger> type*

You should include the following methods in your *Train* class:

* 1. Implement a default constructor which initializes *passengerList* instance variable to a new *ArrayList<Passenger>* object
  2. A method that will take a Passenger object as a parameter and add the passenger to the *passengerList.*
  3. A method returning the reference to the *Passenger* Object at a specific index position. The index position with be passed as a parameter to the method.  
     Hint: The method header would be

*public Passenger getPassenger(int i)*

*{*

*Code goes here*

*}*

* 1. A method returning the number of passengers in the train.
  2. A method returning the percentage of passengers traveling in the first class
  3. A method taking two parameters representing the price of traveling in first and second class and returning the total revenue for the train.
  4. A method checking if a certain person is on the train; if he/she is, the method returns *true;* otherwise, it returns *false.*
  5. Implement the *toString( )* and *equals ( )* method.
  6. Include JavaDoc comments for both Passenger and Train class.

Next you will implement a test class called *TrainTest* with the *main ( )* method. In the main method,

* Create a *Train* object (At this time the train is empty i.e. the passenger List is zero).
* Provide code to read values representing the Passenger information from a text file called *passengers.txt (that you would create).* You can assume that *passengers.txt* has the following format:

<name1> <class1>

<name2> <class2>

….

For instance, the file could contain:

James 1

Ben 2

Suri 1

Sarah 1

Jane 2

…..

* As you read each passengers information, create a Passenger object and add it to the Train.
* Use for loop to print the names of all the passengers in the train.
* Demonstrate the various method of the Train class.
* Make sure to keep your output verbose.

*Please note, the passenger information will be read from a text file in the client program TrainTest and as you read each passenger’s information, you will create a passenger object and add it to the Train object.*

*You will set the orientation of your assignment to Landscape. Type the names of the group members at the top of the assignment along with the assignment #. Include answers to the questions above along with source codes (both user defined classes as well as client programs) in Courier New font. Provide screen shots of the source code in NetBeans for each of the programs you created. Include the output of at least two runs of each of the programs.*