Lab Assignment -3 II MCA OOP Lab

1. A personal phone directory can store first names and phone numbers for maximum 20 people. Assign names and phone numbers for the first 10 people. Prompt the user for a name, and if the name is found in the list, display the corresponding phone number. If the name is not found in the list, prompt the user for a phone number, and add the name and phone number to the list. Continue to prompt the user for names until the user enters “quit”. After the arrays are full containing 20 names, do not allow the user to add new entries.
2. Create a class named TaxPayer. Data fields for TaxPayer include Social Security number(use an int for the type, and do not use dashes within the Social Security number) and yearly gross income. Methods include a constructor that requires values for both data fields, and two methods that each return one of the data field values. Write an application named UseTaxPayer that declares an array of 10 TaxPayer objects. Set each Social Security number to highest possible int and each gross income to zero. Display the 10 TaxPayer objects. Save the files as TaxPayer.java and UseTaxPayer.java. Modify our UseTaxPayer application so each TaxPayer has a successive Social Security number from 1 through 10 and a gross income that ranges from $10,000 to $100,000, increasing by $10,000 for each successive TaxPayer. Save the files as UseTaxPayer2.java
3. Create a class named LibraryBook that contains fields to hold methods for setting and getting a LibraryBooks title, author, and page count. Save the file as LibraryBook.java. Write an application that instantiates five LibraryBook objects and prompts the user for values for the data fields. Prompt the user to enter which field the LibraryBooks should be sorted by – title, author, or page count. Perform the sort procedure and display the LibraryBook objects. Save the file as LibraryBookSort.java.
4. Create a class named Vehicle that acts as a superclass for vehicle types. The vehicle class contains private variables for the number of wheels and the average number of miles per gallon. The Vehicle class also contains a constructor with integer arguments for the number of wheels and average miles per gallon, and a method that returns a String containing these values. Create two subclasses, Car and MotorCycle, that extend the Vehicle class. Each subclass contains a constructor that accepts the miles per gallon value as an argument and forces the number of wheels to the appropriate value ---- 2 for a MotorCycle and 4 for a Car. Write a UseVehicle class to instantiate the two Vehicle objects and print the objects values. Save the files as Vehicle.java, Car.java, MotorCycle.java and UseVehicle.java
5. Create a class named Order that performs order processing of a single item. The class has five fields: customer name, customer number, quantity ordered, unit price, and total price. Include set and get methods for each field except the total price field. The set methods prompt the user for values for each field. This class also needs a method to compute the total price and a method to display the field values. Create a subclass named ShippedOrder that overrides conputePrice() by adding a shipping and handling charge of $4.00. Write an application named UseOrder that instantiates an object of each of these classes. Prompt the user for data for the Order object, and display the results, then prompt the user for data for the ShippedOrder object, and display the results. Save the files as Order.java, ShippedOrder.java and UseOrder.java.
6. Create a class named Book that contains data fields for the title and number of pages. Include get and set methods for these fields. Next, create a subclass named Textbook, which contains an additional field that holds a grade level for the Textbook and additional methods to get and set the grade level field. Write an application that demonstrates using objects of each class. Save the files as Book.java, Textbook.java and DemoBook.java.
7. Create a class named HotelRoom that includes an integer filed for the room number and a double field for the rental rate. Include get methods for these fields and a constructor that requires an integer argument representing the room number. The constructor sets the room rate based on the room number, rooms numbered 299 and below are $69.95 per day and others are $89.95 per day. Create an extended class named Suite whose constructor requires a room number and adds a $40 surcharge to the regular hotel room rate, which again is based on the room number. Write an application named UseHotelRoom that creates an object of each class, and demonstrate that all the methods work correctly. Save the files as HotelRoom.java, Suite.java and UseHotelRoom.java.
8. Write an interface called Numbers with a method int process(int x,int y). Write a class called Sum in which the process() method finds the sum of two numbers and returns an int value. Write another class called Average in which the process() method finds the average of the two numbers and returns an int value.
9. Write an application that prompts the user to enter a number to use as an array size, and then attempt to declare an array using the entered size. If the array is created successfully, display an appropriate message. Java generates a NegativeArraySizeException if you attempt to create an array with a negative size, and a NumberFormatException if you attempt to create an array using a non-numeric values for the size. Use a catch block that executes if the array size is non-numeric or negative, displaying a message that indicates the array was not created.
10. Create an EmployeeException class whose constructor receives a String that consists of an employee’s ID and pay rate. Save the file as EmployeeException.java. Create an Employee class with two fields, idNum and hourlyWage. The Employee constructor requires values for both fields. Upon construction, throw an EmployeeException if the hourlyWages is less than $6.00 or over $50.00. Save the class as Emplyee.java. Write an application that establishes at least three Employees with hourlyWages that are above, below, and within the allowed range. Display an appropriate message when an Employee is successfully created and when one is not. Save the file as ThrowEmployee.java