UNIVERSIDAD DE DAGUPAN

SCHOOL OF INFORMATION TECHNOLOGY EDUCATION

ITP03 / CSP04 | OBJECT ORIENTED PROGRAMMING

Laboratory Manual

Perform the following Laboratory Exercises and Machine Problems.

Objective:

a. To create a Java program using Procedural Approach (EXER1.java).
b. To create a Java program using OOP Approach (EXER2.java).
    1. Creating Classes and Objects:
    2. Create custom Java classes.
    3. Instantiate objects from those classes.
    4. Explain the role of the new keyword in object creation.
    5. Define instance variables (fields) within a class.
    6. Create and use methods within a class.
    7. Describe the purpose of constructors.
    8. Explain the concepts of encapsulation and data hiding.
    9. Understand how classes promote reusability and maintainability in code.
    10. Explain the concepts of encapsulation and data hiding.
    11. Understand how classes promote reusability and maintainability in code.

EXER1.java

Open a New Project in Java as EXER1.java (using Procedural Approach) that will generate results of:

a. addition
b. subtraction
c. multiplication
d. division

When values for the 2 integer numbers as firstNumber and secondNumber is given, respectively.

Consider the I/O Layout shown below:

Enter First Number: _
Enter Second Number: _

The SUM is: _____
The DIFFERENCE is: _____
The PRODUCT is: _____
The QUOTIENT is: _____

NOTE:

The method SUM() returns the value of firstNumber + secondNumber
The method DIFFERENCE() returns the value of firstNumber - secondNumber
The method PRODUCT() returns the value of firstNumber * secondNumber
The method QUOTIENT() returns the value of firstNumber / secondNumber

EXER2.java

1. Open a New Project in Java as EXER2.java (using OOP Approach) that will generate the given I/O Layout.
2. Open a Calculator class, with the followinng requirements:
    a. It has 2 private attributes as firstNumber and secondNumber.
    b. It has a constructor with 2 parameters.
    c. It has 4 regular methods as follows:
      * SUM() which computes for the sum of the 2 numbers. And generate the result.
      * DIFFERENCE() which computes for the difference of the 2 numbers. And generate the result.
      * PRODUCT() which computes for the product of the 2 numbers. And generate result.
      * QUOTIEN() which computes for the quotient of the 2 numbers. And generate result.

NOTE:

The method SUM returns the value of firstNumber + secondNumber
The method DIFFERENCE returns the value of firstNumber - secondNumber
PRODUCT is firstNumber * secondNumber
QUOTIENT is firstNumber / secondNumber

3. The main program will have the following details:
  * import the Scanner class
  * Instantiate the Scanner class as SC.
  * Display and Accept values as shown in the INPUT Layout.
  * Instantiate the Calculator class as myCalc along with the entered values as parameters.
  * Call the methods as shown in the OUTPUT Layout.

4. Consider the I/O Layout shown below:

Enter First Number: _
Enter Second Number: _

The SUM is: _____
The DIFFERENCE is: _____
The PRODUCT is: _____
The QUOTIENT is: _____

Objective:

a. Creating Classes and Objects:
b. Create custom Java classes, using encapsulation. And constructors, and regular methods.

MP1.java

1. Open a New Project in Java as MP1.java (this is the main program).
2. Open a New Java Class as PERSON with the following requirements:
    a. It has 3 attributes: lastName, firstName and Age.
    b. It has a blank constructor and another one with 3 parameters.
    c. It has a displayDetails() method and generates the output below as:

        Hi, my first name is <firstName>.
        And my last name is <lastName>.
        I'm <Age> years old.

3. In the main() program:
a. Import the Scanner class.
b. Instantiate the Scanner class as SC.
C. Enter values as shown in the I/O layout.
b. Instantiate a PERSON class and pass the entered values as parameters.
c. Call the displayDetails() method to generate the result.

4. The I/O Layout:

<<< Data Entry >>>
-
Enter a value for Last Name : _____
Enter a value for First Name: _____
Enter a value for Age: _____
-
-
<<< Personal Details >>>
Hi, my first name is <firstName>.
And my last name is <lastName>.
I'm <Age> years old.

Objectives:
a. Utilize objects to model real-world entities.
b. Implement object interactions in Java programs.

EXER3.java
1. Open a new Project in Java as EXER3.java.
2. Create a PERSON class. Consider the following requirements.
a. The PERSON class has 3 attributtes:
  * name
  * Salary
  * sex code
b. The PERSON class has a blank constructor and a constructor with 3 parameters.
c. The PERSON class has the following regular methods:
  * sexValue() method which tests for the sex code and output the message sex as either "Male" or "Female".
  * Results() method which output the following details:
    Hi, <name>.
    You're doing good with your salary at <salary>.
    And your Sex Code is <sexCode>.

3. The main program will have the following details:
  * import the Scanner class
  * Instantiate the Scanner class as SC.
  * Display and Accept values as shown in the INPUT Layout.
  * Instantiate the PERSON class as P1 along with the entered values as parameters.
  * Call the methods as shown in the OUPUT Layout.

4. Consider the I/O Layout as shown below:

<<< Data Entry >>>
Name: _____
Salary: _____
Sex Code [F/M]: _____
-
-
<<< Person Details >>>
Hi, <name>.
You're doing good with your salary of <salary>.
And your Sex Code is <sexCode>.

EXER4.java

1. Open a New Project in Java as EXER4.java.
2. Open a New Java class as PERSON. Copy & paste and MODIFY the attributes in EXER3.java.
    - It has a sexValue() method which tests for the sex code and output the message sex as either "Male" or "Female".
    - It has a a TaxCalc() method which computes for the tax as 10% of the salary.
    - It has a Results() method which output the following details:

        Hi, <name>.
        You're doing good with your salary at <salary>.
        And your prevailing Income Tax at 10% of your Gross Salary is: <tax>.
        And you're a <sex>.

Objective:

a. Creating Classes and Objects:
b. Create custom Java classes, using encapsulation.
c. Use constructors, Setters and Getters and regular methods.

MP2.java

1. Open a New Project in Java as MP2.java (this is the main program).
2. Open a New Java Class as PERSON (Follow the format in MP1.java).
3. Improve the PERSON class by adding Setters and Getters.
4. Update also the main program.

Objectives:
a. Write Java code to create and manipulate objects.
b. Solve problems by modeling them with classes and objects.
c. Apply inheritance of an object from its super class.
d. Overriding a method from its super class.

MP3.java
1. Open a New Project in Java as MP2.java.
2. Open a New Java Class as PERSON. It has the following details:
   * It has 3 attributes:
      a. Name (ex. Arnaldy D. Fortin)
      b. Sex (ex. Male)
      c. Age (ex. 38)
   * It has a Heading() method, and it displays the following:
      Welcome to Universidad de Dagupan
      <<< DATA ENTRY >>>

* Declare setters and getters.

3. Open a New Java Class as STUDENT. It has the following details:
   * It inherits the attributes of the PERSON class.
   * It has 4 attributes:
      a. Degree (ex. BSCS)
      b. Year (ex. 1)
      c. No. of Units Enrolled (ex. 21)
      d. Miscellaneous Fee (fix value at 4750.00)

   * It has a blank constructor and another one with 6 parameters.
   * It has the following methods:
      a. yearInWords() method returns the equivalent String value based from the Year.

Year	Words
1	First Year
2	Second Year
3	Third Year
4	Fourth Year

      b. tuitionFee() method returns the product of No. of Units Enrolled and 500.
      c. totalFees() method returns the sum of Miscellaneous Fee and tuitionFee().

   * Declare setters and Getters.

4. The main class has the following details:
   * Import a Scanner class.
   * Instantiate a Scanner class as Kbd.
   * Instantiate a STUDENT class as Stud.
   * Display & Accept values based from the I/O Layout below.
   * Generate the output based from the I/O Layout below.

I/O Layout:

Welcome to Universidad de Dagupan
<<< DATA ENTRY >>>
1] Enter Name: ___
2] Enter Sex : ___
3] Enter Age : ___
4] Enter Degree: ___
5] Enter Year[1,2,3,4]: ___
6] Enter No. of Units Enrolled: ___
-
-
1] Name: ___
2] Sex: ___
3] Age: ___
4] Degree: ___
5] Year (Words): ___
6] Units Enrolled: ___
7] Tuition Fee: ___
8] Total Fees: ___

Objective:

1. To create classes and objects using Abstraction.

EXER5.java

1. Open a New Project in Java as EXER5.java (main program).
2. Open a New Java class as ANIMAL.
   a. It has the following attributes:
      - Animal name (**Name**)
      - Animal color (**Kolor**)
      - Animal legs (**NoofLegs**)
   b. It has an abstract method as AnimalSound().
   c. It has a blank constructor. And another one with 3 parameters.
   d. Use setters and getters

3. Open a New Java class as DOG.
   a. It inherits from its super class.
   b. It has an attribute as **dogBreed.**
   c. It a blank constructor and another one which borrows the from the super class.
   d. It has implementation of the AnimalSound() method specific for the DOG class. And the output of which is as follows:

      The DOG says… Arf! Arf! Arf!

4. The main program has the following requirements.
   a. Import the Scanner class
   b. Instantiate a scanner variable as SC.
   c. Instantiate an object of type DOG, as Chihuahua.
   d. Follow the I/O Layout as shown below.

I/O Layout:

<<< DOG DATA ENTRY >>>
-
1] Dog Breed (pls. specify): _____
2] Dog Name: _____
3] Dog Color: _____
4] No. of Legs: _____
-
<<< DOG DETAILS >>>
The dog's name is <Name>.
Its breed is <dogBreed>.
It is color <Kolor>.
And it has <NoofLegs> legs.

UPDATE EXER5.java

5. Open a New Java class as BIRD.
   e. It inherits from its super class.
   f. It has an attribute as **birdBreed.**
   g. It a blank constructor and another one which borrows the from the super class.
   h. It has implementation of the AnimalSound() method specific for the BIRD class. And the output of which is as follows:

      The BIRD says… Chirp! Chirp! Chirp!

Objective:

1. To create classes and objects using Abstraction along with Array.

abstractEmployee.java

Write a Java program to generate the following requirements.

1. **Create a new Project: abstractEmployee**
2. **Create a new Abstract class called Employee**
   - **a.** It has two (3) private attributes ***empName, yrsofService*** and ***Salary***, each of which is a String, double and double respectively.
   - **b.** It has a blank constructor as well as another constructor with 2 parameters for the ***empName*** and ***yrsofService***.
   - **c.** It contains two (2) abstract methods: ***bonusCalculate()***, ***grossSalary()***
   - **d.** It has a regular method as ***displayInfo()***.
   - **e.** It has setters and getters.

3. **Create a subclass called Supervisor that extends the Employee class.**
   - **a.** It has a private attribute ***salaryBonus***, and ***serviceBonus*** of type double.
   - **b.** It has a blank constructor and another one with parameters from the parent class.
   - **c.** It has its own ***BonusSalary()*** method that returns the value of the ***salaryBonus*** based on the following conditions:
     - When the ***salary*** is >=30,000. ***salaryBonus*** is equivalent to **40*% of the salary*** otherwise ***salaryBonus*** is equivalent to ***20% of the salary***.
   - **d.** It has its own ***BonusService()*** method that returns the value of the ***serviceBonus*** based on the given formula:
     - ***serviceBonus*** is equivalent to ***yrsofService*** * **500**;
   - **e.** ***bonusCalculate()*** method returns the **SUM** of the ***salaryBonus*** *and* ***serviceBonus***
   - **f.** The ***grossSalary()*** method returns the value of the total amount which is based from ***Salary*** and ***SUM*** of all ***Bonuses***.
   - **g.** The ***displayInfo()*** method simply outputs the following details:

     Manager Name: _____
     Salary: _____
     Salary Bonus: _____
     No. of years in service: _____
     Service Bonus: _____
     Total Amount: _____
   - **h.** It has a setter and getter.

4. **The main program (Abstract_Employee) has the following details:**

```
//import a Scanner
//after the public static void main (String[] args){
        //instantiate a Scanner variable as SC
        //instantiate a Supervisor class as SV
        //display <<< Data Entry for Supervisor >>>
        //display and accept an input for the following
        //      Name: __
        //      Salary: __
        //      No. of Years in Service: __

        //display <<< Supervisor Details Report >>>

        //call displayInfo() from Supervisor class
```

```
Run:
<<< Data Entry for Supervisor >>>
Name: ARNALDY D. FORTIN
Salary: 20000
No. of Years in Service: 30

<<< Supervisor Details Report >>>
Supervisor Name: ARNALDY D. FORTIN
Salary: 20000.0
Salary Bonus: 4000.0
No. of Years in Service: 30.0
Service Bonus: 15000.0
Total Bonus: 19000.0
```

MP3.java

Write a Java program to generate the following requirements.

1. Create a new Project: MP3.java
2. Create a new Abstract class called Employee. Copy & Paste the contents of the Employee class from the abstractEmployee.java.
3. Create a subclass called Manager that extends the Employee class. Copy & Paste the contents of the Employee class from the abstractEmployee.java.
4. The main program has the following details:

```
//import a Scanner
//after the public static void main (String[] args){
        //instantiate a Scanner variable as SC
        //instantiate a Manager class as M and an Array of 5 managers.

        //display <<< Data Entry for Supervisor >>>

        //display and accept 5 inputs for the following
        //Entry No: 1
        //      Name: __
        //      Salary: __
        //      No. of Years in Service: __
        //Entry No: 2
```

```
//        Name: __
//        Salary: __
//        No. of Years in Service: __
//Entry No: 3
//        Name: __
//        Salary: __
//        No. of Years in Service: __
//Entry No: 4
//        Name: __
//        Salary: __
//        No. of Years in Service: __
//Entry No: 5
//        Name: __
//        Salary: __
//        No. of Years in Service: __

//display <<< Manager Details Report >>>
//Generate the results for each of the 5 Managers
//call displayInfo() from Manager class
```

Objective:

2. To create classes and objects using Interface.

EXER6.java

1. Open a New Project in Java as EXER6.java (main program).
2. Open a new Interface as landAnimal.
    - It has abstract method as LandDetails().
3. Open a New Java class as ANIMAL.
    - Copy & Paste the contents of the ANIMAL class from **EXER5.java**.

4. Open a New Java class as DOG and inherits from ANIMAL and landAnimal classes.
    - Copy & Paste the contents of the DOG class from **EXER5.java**.
    - It will execute the LandDetails() as:

      <Name> is a Land Animal.
      All Land Animal can walk, run, and jump.

6. The main program has the following requirements.
    a. Import the Scanner class
    b. Instantiate a scanner variable as SC.
    c. Instantiate an object of type DOG, as Chihuahua.
    d. Follow the I/O Layout as shown below.

I/O Layout:

<<< DOG DATA ENTRY >>>
-
1] Dog Breed (pls. specify): _____
2] Dog Name: _____
3] Dog Color: _____
4] No. of Legs: _____
-
<<< DOG DETAILS >>>
The dog's name is <Name>.
Its breed is <dogBreed>.
It is color <Kolor>.
And it has <NoofLegs> legs.

-
<Name> is a Land Animal.
All Land Animal can walk, run, and jump.

UPDATE EXER6.java

5. Open a new Interface as airAnimal.
    - It has abstract method as airDetails().
    - It will execute the airDetails() as:

      <Name> is an Air Animal.
      All Air Animal can walk, run, jump, fly and dive.

7. Open a New Java class as BIRD.
    a. It inherits from its super class ANIMAL and from airAnimal.
    b. It has an attribute as **birdBreed.**
    c. It a blank constructor and another one which borrows the from the super class.
    d. It has implementation of the AnimalSound() method specific for the BIRD class. And the output of which is as follows:

       The BIRD says… Chirp! Chirp! Chirp!

I/O Layout:

<<< BIRD DATA ENTRY >>>
-
1] Bird Breed (pls. specify): _____
2] Bird Name: _____
3] Bird Color: _____
4] No. of Legs: _____
-
<<< BIRD DETAILS >>>
The bird's name is <Name>.
Its breed is <dbirdBreed>.
It is color <Kolor>.
And it has <NoofLegs> legs.
-
<Name> is an Air Animal.
All Air Animal can walk, run, jump, fly and dive.

EXER7.java

Write a Java program to generate the following requirements.

1. Open a New Project in Java as EXER7.java (main program).
2. Generate the given I/O Layout using try-catch-finally block.
3. Place the try-catch-finally block inside a do-while loop statement.
4. Convert any response for Try Again [Y/N] in UpperCase format.

I/O Layout:

```
run:
Enter an Integer Number: 55

That is correct!

Try Again [Y/N]? y

**********

Enter an Integer Number: a

That is incorrect!

Try Again [Y/N]? y

**********

Enter an Integer Number: 199

That is correct!

Try Again [Y/N]? n

**********

BUILD SUCCESSFUL (total time: 20 seconds)
```

EXER8.java

Write a Java program to generate the following requirements.

1. Open a New Project in Java as EXER8.java (main program).
2. Generate the given I/O Layout using try-catch block for the 1st Integer Number.
3. Then use try-catch-finally for the 2nd Integer Number. Wherein the finally block, test if the 2nd Number is 0, if yes, output:

   Cannot Divide 1st Number, By 2nd Number.

   Otherwise:
   Compute for Q which is the quotient of Num1/Num2.
   Then display the message:
   "1st Num "+Num1+" Divided By 2nd Num "+Num2+" IS EQUAL TO "+Q

4. Place the try-catch and try-catch-finally blocks inside a do-while loop statement.
5. Convert any response for Try Again [Y/N] in UpperCase format.

I/O Layout:

```
run:
Enter First Integer Number: a

That is incorrect!

Enter Second Integer Number: b

That is incorrect!

Cannot Divide 1st Number 0 By 2nd Number 0

Try Again [Y/N]? y

**********

Enter First Integer Number: 55

That is correct!

Enter Second Integer Number: a

That is incorrect!

Cannot Divide 1st Number 55 By 2nd Number 0

Try Again [Y/N]? y

**********

Enter First Integer Number: 5

That is correct!

Enter Second Integer Number: 0

That is correct!

Cannot Divide 1st Number 5 By 2nd Number 0

Try Again [Y/N]? n

**********

BUILD SUCCESSFUL (total time: 44 seconds)
```

IO_STREAMING1.java

Write a Java program to generate the following requirements.

1. Open a New Project in Java as IO_STREAMING1.java (main program).
2. Generate the given I/O Layout

I/O Layout:

```
run:
<<< FILE MANIPULATION MENU >>>

1] Create a File --- sampleko.txt
2] Write Data in the File --- sampleko.txt
3] Read Data in the File --- sampleko.txt
4] Delete the file --- sampleko.txt

Ano Pipiliin Mo [1/2/3/4]? 1

************************

File created: sampleko.txt
Absolute path: d:\myOOP\sampleko.txt
BUILD SUCCESSFUL (total time: 1 minute 1 second)


run:
<<< FILE MANIPULATION MENU >>>

1] Create a File --- sampleko.txt
2] Write Data in the File --- sampleko.txt
3] Read Data in the File --- sampleko.txt
4] Delete the file --- sampleko.txt

Ano Pipiliin Mo [1/2/3/4]? 2

************************

Enter your Last Name:
FORTIN
Enter your First Name:
ARNALDY
Successfully wrote to the file.
BUILD SUCCESSFUL (total time: 15 seconds)


run:
<<< FILE MANIPULATION MENU >>>

1] Create a File --- sampleko.txt
2] Write Data in the File --- sampleko.txt
3] Read Data in the File --- sampleko.txt
4] Delete the file --- sampleko.txt

Ano Pipiliin Mo [1/2/3/4]? 3

************************

Reading the contents...

FORTIN, ARNALDY
BUILD SUCCESSFUL (total time: 4 seconds)
```

```
run:
<<< FILE MANIPULATION MENU >>>

1] Create a File --- sampleko.txt
2] Write Data in the File --- sampleko.txt
3] Read Data in the File --- sampleko.txt
4] Delete the file --- sampleko.txt

Ano Pipiliin Mo [1/2/3/4]? 2

*************************

Enter your Last Name:
FUKIEKOW
Enter your First Name:
MAE KHOTOW
Successfully wrote to the file.
BUILD SUCCESSFUL (total time: 31 seconds)


run:
<<< FILE MANIPULATION MENU >>>

1] Create a File --- sampleko.txt
2] Write Data in the File --- sampleko.txt
3] Read Data in the File --- sampleko.txt
4] Delete the file --- sampleko.txt

Ano Pipiliin Mo [1/2/3/4]? 3

*************************

Reading the contents...

FORTIN, ARNALDY
FUKIEKOW, MAE KHOTOW
BUILD SUCCESSFUL (total time: 4 seconds)


run:
<<< FILE MANIPULATION MENU >>>

1] Create a File --- sampleko.txt
2] Write Data in the File --- sampleko.txt
3] Read Data in the File --- sampleko.txt
4] Delete the file --- sampleko.txt

Ano Pipiliin Mo [1/2/3/4]? 4

*************************

Sure ka, mai ERASE lahat ng data sa File [Y/N]? Y
BUILD SUCCESSFUL (total time: 13 seconds)
```

```
run:
<<< FILE MANIPULATION MENU >>>

1] Create a File --- sampleko.txt
2] Write Data in the File --- sampleko.txt
3] Read Data in the File --- sampleko.txt
4] Delete the file --- sampleko.txt

Ano Pipiliin Mo [1/2/3/4]? 3

************************

An error occurred. Kasi na-DELETE na ang file kanina.
BUILD SUCCESSFUL (total time: 9 seconds)
```

IO_STREAMING2.java

Write a Java program to generate the following requirements.

3. Open a New Project in Java as IO_STREAMING2.java (main program).
4. Copy and Paste from IO_STREAMING1.java.
5. Replace the filename with myfile.csv.
6. Generate the given I/O Layout

I/O Layout:

```
run:
<<< FILE MANIPULATION MENU >>>

1] Create a File --- myfile.csv
2] Write Data in the File --- myfile.csv
3] Read Data in the File --- myfile.csv
4] Delete the file --- myfile.csv

Ano Pipiliin Mo [1/2/3/4]? 1
```

NOTE: The Filename is myfile.csv. Perform #1. Then #2 at least 3 times. Enter the following details:

```
ZINUZOWKU
ZUSOWMOU

LAILAINAH
DEDEKOU

COLINTHS
JAHJAH
```

Once done with the three entries… Locate the file in your computer and **DOUBLE CLICK IT**.

What have you observed???