LAB MANUAL

Object Oriented Programming

B.Tech. Semester-II

Session: 2022-23

Shri Ramdeobaba College of Engineering and Management,
Gittikhadan,Katol Road,
Nagpur 440013 (M.S.)

Course Objectives
1. To develop ability of students to implement basic concepts and techniques of object-oriented programming paradigm like encapsulation, inheritance, polymorphism, exception handling.
2. Develop solution to problems using collection classes, generics, streams, multithreading.

Course Outcomes
On completion of the course the student will be able to
1. Design solution to problems using concepts of object-oriented programming like classes, objects, inheritance with proper exception handling.
2. Use collection classes, generic classes to design programs.
3. Implement programs based on streams and multithreading.

Syllabus
Experiments based on the syllabus of the theory course.

Teaching Scheme
Load per week: 01 (2 hours slot)
Credits: 01

Evaluation Scheme:
Internal Assessment : 25 marks (Marks distribution is declared in TA Plan)
End Semester Examination : 25 marks

Hardware / software requirements

Hardware:
Free Disk Space: A minimum of 2.5 gigabytes
CPU: 1 gigahertz or better
RAM: 1 gigabyte or more

Software:
Operating System: Ubuntu
Java Version: 12.0 or greater (LTS)
Editor: gedit, Visual Studio Code

List of Experiments

SN	Aim/Problem Statement	Mapped COs
1	Create a class complex and provide functionalities to add and multiply 2 complex numbers. Class Complex will have real and imaginary as the data members. Write an appropriate main() method to demonstrate the functionalities.	CO1
2	Write a program to create a class OnlineShoppingPortal. The shopping portal has customers of two categories, Prime and Regular. The checkout functionality for both the types of customers needs to be designed as follows. "Prime" customer requires only amount to be paid for checkout and "Regular" customer must provide amount and promocode. A prime customer gets 30% off if cart value > 1000, else 20% off. A regular customer gets only Rs.10 off if cart value < 500, else 10% off (provided that the code matches "GET10"). Demonstrate method overloading on a method checkout() which will display the original and discounted amount. Create an object in main() to demonstrate method overloading.	CO1
3	A. Create a class Time with the data members as hours and minutes. Add Functionality to add and subtract 2 time objects. Test the time class is main(). B. "GreatClock" (A scientific research company) wants advanced time objects which will also provide functionalities of addition and subtraction of seconds and milliseconds along with hours and minutes. How will you add this feature without changing the Time Class? C. Create a class TimeZone which will add functionality to convert the time from one time zone to another time zone. Note: Class Time zone uses the Time Object and uses the Add and Subtract methods of Time Class.	CO1
4	A web application has a class User to represent basic user's information like username, password, mobile number, isActive,gender and created date and a method printUser() which prints the basic information in a specific format that should not be changed by the inheriting classes. There are two types of Users viz. Standard user and custom user. A standard user has additional fields like alias,last login and role whereas a custom user has fields like email, Security Key and Manager. Use the inheritance	CO1

	feature of OOP to implement this system (use super,final,constructors,etc. as required). Define methods to display appropriate information based on the user type. Create an array of users and display information of all users based on following criteria: i) Display all female standard users ii) Display all custom users created before 1 Jan 2000 with email Id containing "gmail.com".	
5	Write a program to implement multiple inheritance. Consider a class BankAccount with data members as account number, aadhar number, owner name, ROI and balance with member functions openAccount(), deposit(amount),closeAccount() and updateInterest(). Create an interface Debitable which has method withdraw(). Derive a class FixedDepositAccount from BankAccount having data member lockInPeriod. Override methods updateInterest() to update Simple Interest, and method closeAccount() to charge 5 % for closure of FD Account before lockInPeriod. Derive a class SavingAccount from class BankAccount and interface Debitable. The account numbers should be serial numbers of 5 digit and automatically assigned on object creation, such that all FDAccounts start with 55 and all Saving account start with 11. [ROI for Saving Account is 4% and for FD – 1-2yrs-6% ; 2-5yrs-6.5% ; >5yrs- 7%]	CO1
6	Write a generic class having a function addArray() to add all elements of the array and print the sum. Write a class to test the generic class and its method. Define a method isEqual() to test whether the sum of two arrays is same or not. Define a user defined exception class 'UnmatchedSum' which is thrown if two sums are not equal. Use this class to throw an exception in the method isEqual() if the sums are different.	CO1, CO2
7	Create a class Product having private data members product_name, cost, manufacturer,max_discount. Include appropriate methods to set the member values and override toString method to display the data members. Create an arraylist to store 10 product objects. Take the input from file using BufferedReader. Write a menu driven program to 1) Display the list of products using iterator 2) Display the list of products whose max_discount is 50%. Also display the final cost at which the product can be given. 3) Products sorted according to the cost 4) Products sorted according to the manufacturer	CO2, CO3

8	Demonstrate the use of multithreading. Consider a website publishes live cricket score. The server thread can change the contents of the website whereas all the client threads read the score. Write the code to demonstrate all the functionalities.	CO3

IMPORTANT GUIDELINES

1) Understand the problem statement thoroughly
2) Design a solution in the form of class diagram and (or) algorithm to solve the given problem using appropriate features and constructs of Java
3) Write a JAVA program to implement the solution design
4) Run the program
5) Check the correctness of your program using the given test cases
6) If the desired output is not obtained, do necessary corrections to the program
7) Write an appropriate conclusion about the methodology used, output obtained, possibility of improvement or optimization and applicability of your program.
8) Write answers to the given review questions

STEPS FOR INSTALLATION OF JAVA ON UBUNTU

1. Go to https://www.oracle.com/in/java/technologies/downloads/
2. Download the Debian package for Linux (this gets downloaded to the Downloads folder)
3. cd Downloads/ (directory which contains java installation .deb file)
4. sudo dpkg -i jdk-12.0.2_linux-x64_bin.deb
5. sudo update-alternatives --install /usr/bin/java java /usr/lib/jvm/jdk-12.0.2/bin/java 1
6. sudo update-alternatives --install /usr/bin/javac javac /usr/lib/jvm/jdk-12.0.2/bin/javac 1
7. java --version
8. javac --version
9. sudo update-alternatives --config java
10. copy path from "/usr... upto JDK12.0.2"
11. sudo gedit /etc/environment
(This opens a file)
12. At the bottom of the file, write the following
JAVA_HOME="Paste the path copied in step 10, in double quotes " and save the file
13. sudo gedit /etc/environment
14. ECHO $JAVA_HOME (To verify if the Java Home Path is set properly)

GUIDELINES FOR PRACTICAL SUBMISSION

Follow the below format for Practical Journal Submission:

1. Problem Statement

2. Design of Solution (Algorithm/Class Diagram/Theory)

3. Code (*Print out*)

4. Test Cases (*Print out)*

5. Conclusion (*Write an appropriate conclusion about the methodology used, output obtained, possibility of improvement or optimization and applicability of your program)*

6. Review questions and answers

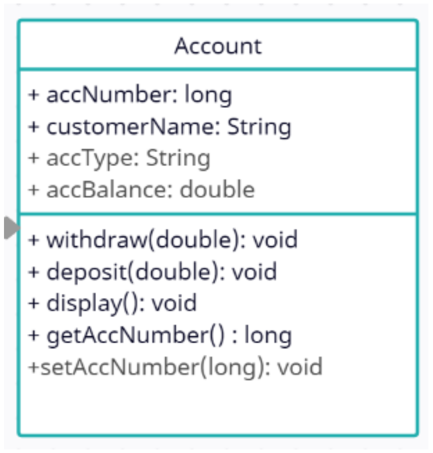
A SAMPLE OF PRACTICAL SUBMISSION FOR REFERENCE

1. AIM:

Define a class Account to represent a bank account information including account number, customer name, account type, and account balance and methods to perform withdraw, deposit and display operations. Create an account object using this class and call appropriate methods.

2. DESIGN OF SOLUTION:

Class Diagram



<<Explain following points in brief>>

1. Selection of datatypes with proper justification
2. Justification for each type of constructor (if applicable)
3. Number of classes required and explanation for each class design (and their relationship, if applicable)
4. Explanation for the types of inputs and assumptions, if any
5. Explanation for the logic design of the important methods (like deposit(), withdraw())
6. Any other important point applicable for the problem statement

3. CODE:

```
class Account{
   long accNumber;
   String customerName;
   String accType;
   double accBalance;
   // Define getter and setter methods
   void withdraw(double amount){
      if(accBalance-amount<0.0){
          System.out.println("Insufficient funds");
      }else{
          accBalance=accBalance-amount;
      }
   }
   void deposit(double amount){
      accBalance=accBalance+amount;
   }
   void display(){
      System.out.println("========Account Information========");
      System.out.println("Account Number   :"+getAccNumber());
      System.out.println("Customer Name    :"+getCustomerName());
      System.out.println("Account Type     :"+getAccType());
      System.out.println("Account Balance  :"+getAccBalance());
   }
}
class TestDrive{
public static void main(String[] args) {
          Account accObj = new Account();
          accObj.accNumber=123456789;
          accObj.customerName="Bunti Modi";
          accObj.accType="Saving";
          accObj.accBalance=5000;
          accObj.deposit(10000);
          accObj.withdraw(500);
          accObj.display();
}
}
```

4. TEST CASES:

Account accObj = new Account();

accObj.accNumber=123456789;

accObj.customerName="Bunti Modi";

accObj.accType="Saving";

accObj.accBalance=5000;

SN	Input	Expected Output	Actual Output
1	deposit(500)	Balance : 5500	Balance : 5500 (printed in display() method)
2	withdraw(2000)	Balance : 3500	Balance : 3500
3	withdraw(4000)	*Insufficient Funds*	*Insufficient Funds*

5. CONCLUSION:

The implementation can be made generalized and lines of code (LOC) can be reduced by using a constructor. The process of generation of account number can be automated. Account type can be represented in a better way using inheritance. Validation can be added to avoid null information during object creation. This program can be used as module in Banking Application to perform basic account operations.

6. REVIEW QUESTIONS:

1. How to extend the program to create multiple accounts?

Ans. Define an array (ArrayList for dynamic size) of Account Objects and use iterations to perform bulk operations.

2. How to count the number of Account objects created?

Ans. Use static data member (counter) and increment the counter in the constructors.

PRACTICAL NO. 1

AIM:

Create a class ComplexNumber and provide functionalities to add and multiply 2 complex numbers. Class ComplexNumber will have real and imaginary as the data members. Write an appropriate main() to demonstrate the functionalities.

TEST CASES:

SN	Input	Expected Output	Actual Output
1	Number 1 = 2 + 5i Number 2 = 4 + 6i	Addition: Number 3 = 6 + 11i Multiplication: Number 3 = -22 + 32i	
2	Number 1 = 8 + -5i Number 2 = 4 + -2i	Addition: Number 3 = 7 + 8i Multiplication: Number 3 = 22 - 36i	
3	Input Complex numbers created with parameter-less constructor	Addition: Number 3 = 0 + 0i Multiplication: Number 3 = 0 + 0i	
4	Number 1 = 1.5 + 4.6i Number 2 = 8.7 + 22.4i	Addition: Number 3 = 10.2 + 27i Multiplication: Number 3 = -89.99 + 73.62i	
5	Any input other than numbers or an input outside the range -100000.0 to +100000.0	Show a message "The input is invalid"	

REVIEW QUESTIONS:

a. How to call a method without a need to create an object?
b. How does your code handle invalid input or blank input?
c. How will you modify your code to handle the results of addition/ multiplication which are beyond the double range?
d. Find out the ways to take end user input from the console/terminal.