Ahmedabad University
School of Engineering and Applied Science
B. Tech. (ICT) Semester – II
CSC103 – Object Oriented Programming Lab
Lab Assignment – 3

Last Date of Submission: To be announced soon

Topics covered: Object Oriented Concepts, Inheritance, Polymorphism, Abstract Class

Important Instructions:

Follow below coding and naming conventions while developing your programs:

1. Class names should be nouns and first letter must be capital letter. The first letter of each internal word capitalized. Use whole words-avoid acronyms and abbreviations E.g. BankAccount, Employee, AdminStaff, SavingAccount
2. Methods should be verbs with camel case, first letter lowercase and the first letter of each internal word capitalized.
   E.g. addEmployee( );   deleteEmployee( );   displayEmployeeList( );
3. Variables: Variable names should be meaningful. E.g. for Employee Class, variable names should be employeeId, employeeName, employeeDesignation, employeeScaleOfPay, etc.
4. Every Program should have header having following information in multi-line comments or java doc comments

```
/*
   @author RollNo. - Firstname Lastname                    @version dd/mm/yyyy
  Description: Write program definition.
*/
```

   Every Program should have footer with output of the Program in multi-line comment

```
/*   PROGRAM OUTPUT  */
```

Policy on Plagiarism (Copying of code/programs) and academic dishonesty

Copying Lab assignment/program code doesn't help you to learn the concepts.

Programs designed, developed and submitted by a student should be the result of original and individual work based on his/her own efforts. Full or part of the code should not be copied from internet or from peer students or other sources. A student should not share/circulate the code/programs developed by them (for individual assignments) with their peers in any form. Violation of above will be considered as academic dishonesty and any such case will be strictly dealt with and liable to get zero in the evaluation.

Exercises based on Inheritance, Polymorphism

This laboratory exercise helps you to understand the *basic concepts* of Object Oriented Programming.

1. Write a program to define a superclass Record which stores the names and ranks of 5 students and a subclass Rank to find the highest rank along with the name.

A super class **Record** has been defined to store the names and ranks of 5 students. Define a sub class **Rank** to find the highest rank along with the name. The details of both classes are given below:

Class name : Record

Data Members:

- **name[ ] :** to store the names of students
- **rnk[ ] :** to store the ranks of students

Member functions:

- **Record() :** constructor to initialize data members
- **void readValues() :** to store names and ranks
- **void display() :** displays the names and the corresponding ranks

Class name : Rank

Data Members:

- **index :** integer to store the index of the topmost rank

Member functions:

- **Rank() :** constructor to invoke the base class constructor and to initialize index to 0.
- **void highest() :** finds the index location of the topmost rank and stores it in index without sorting the array
- **void display() :** displays the name and ranks along with the name having the topmost rank.

Using the concept of inheritance, define the class **Rank** to fulfill the required result.

****

2. Write a program to design a superclass Detail which stores the details of a customer and a subclass Bill which calculates the telephone bill.

Hint: Define a super class **Detail** to store the details of a customer as described below. Define a subclass **Bill** to compute the monthly telephone charge of the customer as per the rules given below:

Number Of Calls	Rate
1 – 100	Only Rental charge
101 – 200	Rs 2 per call + rental charge
201 – 300	Rs 3 per call + rental charge
Above 300	Rs 5 per call + rental charge

The details of both the classes are given below:

Class Name : Detail

Data members / Instance variables:

- **name :** to store the name of the customer.
- **address :** to store the address of the customer.
- **telno :** to store the phone number of the customer.
- **rent :** to store the monthly rental charge

Member functions:

- **Detail(…) :** parameterized constructor to assign values to data members.
- **void show() :** to display the detail of the customer.

Class Name : Bill

Data members / Instance variables:

- **n :** to store the number of calls.
- **amt :** to store the amount to be paid by the customer.

Member functions:

- **Bill(…) :** parameterized constructor to assign values to data members of both classes and to initialize amt = 0.0.
- **void cal() :** calculates the monthly telephone charge as per the charge given above.
- **void show() :** to display the detail of the customer and amount to be paid.

Using the concept of inheritance, define the class **Bill** to fulfill the giving details.

****

3. Write a Java program to design a superclass Perimeter which calculates the perimeter of a parallelogram. Define a subclass Area which calculates the area of a parallelogram by using the required data members of the super class.

The class details are as under:

Class name : Perimeter

Data members/instance variables:

- **a** : to store the length in decimal
- **b** : to store the breadth in decimal

Member functions:

- **Perimeter(…)** : parameterized constructor to assign values to data members
- **double Calculate( )** : calculate and return the perimeter of a parallelogram as 2 * (length + breadth)
- **void show()** : to display the data members along with the perimeter of the parallelogram

Class name : Area

Data members/instance variables:

- **h** : to store the height in decimal
- **area** : to store the area of the parallelogram

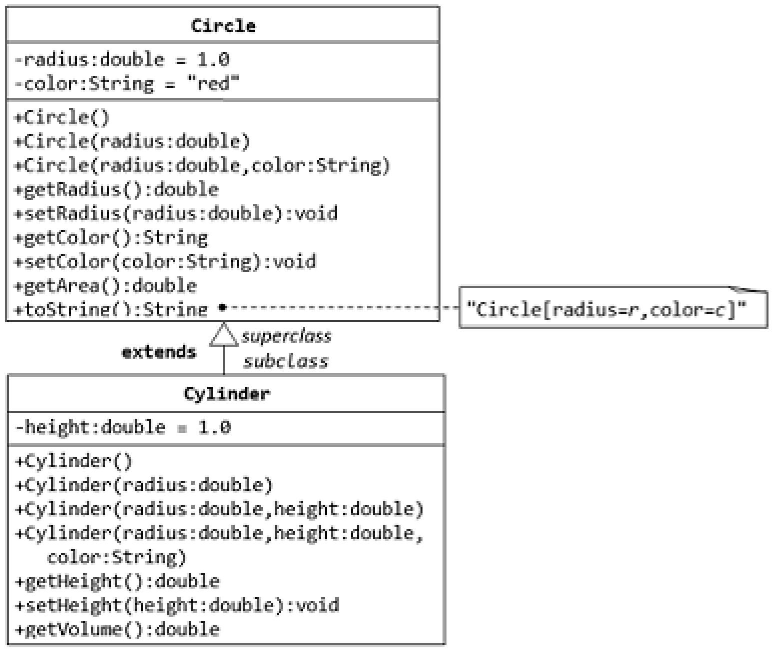
Member functions:

- **Area(…)** : parameterized constructor to assign values to data members of both the classes
- **void doarea( )** : compute the area as (breadth * height)
- **void show()** : display the data members of both classes along with the area and perimeter of the parallelogram.

****

4. Exercise related to the Circle and the Cylinder Class

A subclass called **Cylinder** is derived from the superclass **Circle** as shown in the following class diagram (where an arrow pointing up from the subclass to its superclass). Study and understand how the subclass Cylinder invokes the superclass' constructors (via super() and super(radius)) and inherits the variables and methods from the superclass Circle. Write a test program (TestCylinder) to test the Cylinder class created.



The subclass Cylinder inherits getArea() method from its superclass Circle. Try overriding the getArea() method in the subclass Cylinder to compute the surface area (=2π×radius×height + 2×base-area) of the cylinder instead of base area. That is, if getArea() is called by a Circle instance, it returns the area. If getArea() is called by a Cylinder instance, it returns the surface area of the cylinder.

****

Exercise based on Abstract Class

5. Create abstract superclass **Figure** having **abstract method double area ( ).** Create **concrete subclasses Rectangle and Circle** of Figure which provide specific implementation of area ( ) method.

- **Rectangle** has length and breadth data-members while circle has radius data-member. Define parameterized constructors in both classes.
- **In main( ),** create objects of Rectangle and Circle classes with suitable length, breadth and radius.
- **Demonstrate use of run-time polymorphism** concepts by invoking the area( ) method of Rectangle and Circle objects.

6. Define an abstract class called **Animal** with two abstract and one concrete method as shown below:

abstract methods

```
abstract void move();
abstract void eat();
```

concrete method

```
void label()
```

Extend the **Animal** abstract class with two child classes: **Bird** and **Fish**. Both of them has their own functionality for the move() and eat() abstract methods. For example display message “Bird moves by flying” in move() method of Bird.

Define TestBird and TestFish classes. Call the concrete (label()) and the abstract (move() and eat()) methods from TestBird and TestFish classes for demonstration purpose.

In one of the main( ) method, create a reference variable of Animal class. Ask the user for choice of his/her pet Animal. Based on user’s choice, create suitable sub-class animal object and invoke the methods of animal using base class variable.

7. **Inheritance exercise: Parent and Child class example**

Create and compile a simple class called **Parent**. Give it the following behaviour:

a. A default constructor that does nothing other than print out “Parent default constructor” using `System.out`
b. A single method called `getMessage` which returns a String, e.g. “Parent message”

Then create and compile a class called **Child**. Give it the following behaviour

a. Do not give it a constructor
b. Override the parent’s `getMessage` method to return an alternative String. E.g. “Child message”

c. A main method which creates an instance of the Child object, and then writes the value returned by its `getMessage` method to the command line.

Alter the Child class to give it a default constructor which prints out “Child default constructor”. Compile and run the application again and identify what happens.

8. **Exercise based on polymorphism: Advertisement (Hoarding, NewspaperAd) example**

An advertising campaign consists of a set of advertisements. Advertisement can be of several types, for example: a roadside Hoarding and a Newsapaper ad. Each type of advertisement has a cost associated with it: a fixed fee (to cover materials, production staff and media costs) and variable costs (for buying advertising time and space).

The cost is calculated as under for each category of advertisement:

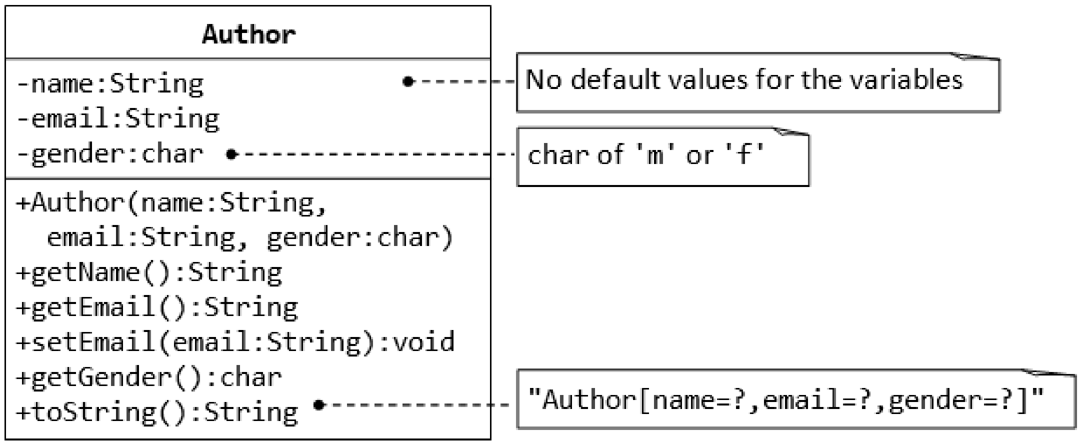
- A Hoarding for a poster is hired for a number of days. The Hoarding Ad cost (Rs. H) per hoarding. [Hint: Ccost for Hoarding (H) = Fixed Fee + No. of Days * RatePerDay].
- Newspaper: The cost is Rs. N per column cm. Total cost of Newpaper Ad is Fixed Fee + N * ColumnCm.

Define a parent class **Advertisement**. Define child/subclasses: Hoarding and NewspaperAd. Defining **AdvertisementTest** class, demonstrate how the cost() methods in classes Hoarding and NewspaperAd are used for calculating total cost of the given category of advertisement.

****

9. Following exercise is based on ArrayList. In Java ArrayList is used to store dynamically sized collection of elements.

Define Author class as per the information given below:


Author
-name:String
-email:String
-gender:char
+Author(name:String, email:String, gender:char)
+getName():String
+getEmail():String
+setEmail(email:String):void
+getGender():char
+toString():String
No default values for the variables
char of 'm' or 'f'
"Author[name=?,email=?,gender=?]"

Demonstrate how ArrayList can be used to store collection of objects of type Authors. Demonstrate how to perform following operations:

- How to add new element in list
- How to remove element from list
- How to check if an ArrayList is empty using the isEmpty() method.
- How to find the size of an ArrayList using the size() method.
- How to access the element at a particular index in an ArrayList using the get() method.
- How to modify the element at a particular index in an ArrayList using the set() method.

****