**Part-A**

1. Write a Java Program that does the following related to **Inheritance**:
   1. Create an **abstract** class called ‘Vehicle’ which contains the 'hashelmet',‘year of manufacture’ and two abstract methods ‘getData()’ and ‘putData()’. Demonstrate the error when attempt is made to create objects of ‘Vehicle’.
   2. Have two **derived** classes ‘TwoWheeler’ and ‘FourWheeler’. ‘FourWheeler’ is a **final** class. Demonstrate the error when attempt is made to inherit from ‘FourWheeler’.
   3. Your abstract class should have overloaded**constructors** that initializes 'hashelmet' and ‘year of manufacture for TwoWheeler and FourWheeler respectively.
   4. ‘TwoWheeler’ has data elements ‘Brand’, ‘Cost’, ‘EngineType’ (possible values “2 stroke”, “4 stroke”), and ‘Color’ which are **private, protected**, **‘friendly/default’** and **public** respectively. Demonstrate the various ways in which the two abstract methods can be dealt ‘getData()’ and ‘putData()’ can be dealt with by the derived classes, ‘TwoWheeler’ and ‘FourWheeler’.
   5. The sub-class of ‘TwoWheeler’ called ‘MyTwoWheeler’ has the element ‘Owner Name’.
   6. Use the **super** keyword to initialize an object of ‘MyTwoWheeler’ with all the values of its parent class data elements.
2. Write a Java Program that does the following:
   1. Create an **abstract** class called ‘Shape’ which contains Two instance variables color (String) and filled (boolean).

* Two constructors: a no-arg (no-argument) constructor that initializes the color to "green" and filled to true, and a constructor that initializes the color and filled to the given values.
* Getter and setter for all the instance variables. By convention, the getter for a boolean variable xxx is called isXXX() (instead of getXxx() for all the other types).
* A toString() method that returns "A Shape with color of xxx and filled/Not filled".
* An abstract method getArea()
* Demonstrate the error when attempt is made to create objects of ‘Shape’.
  1. Write two **subclasses** of Shape called Circle and Rectangle. Rectangle is a **final** class. Demonstrate the error when attempt is made to inherit from ‘Rectangle’.
  2. Write a class called Square, as a subclass of Rectangle. Convince yourself that Square can be modeled as a subclass of Rectangle. Square has no instance variable, but inherits the instance variables width and length from its superclass Rectangle.

1. Write a Java Program that does the following
2. Create a superclass, Student, and two subclasses, Undergrad and Grad.
3. The superclass Student should have the following data members: name, ID, grade, age which are **private, protected**, **‘friendly/default’** and **public** respectively.
4. The superclass, Student should have at least one method: booleanisPassed (double grade)
5. The purpose of the isPassed method is to take one parameter, grade (value between 0 and 100) and check whether the grade has passed the requirement for passing a course. In the Student class this method should be empty as an abstract method.
6. The two subclasses, Grad and Undergrad, will inherit all data members of the Student class and override the method isPassed. For the UnderGrad class, if the grade is above 70.0, then isPassed returns true, otherwise it returns false. For the Grad class, if the grade is above 80.0, then isPassed returns true, otherwise returns false.
7. Demonstrate **"final"** keyword in the above class.
8. Create a test class for your three classes. In the test class, create one Grad object and one Undergrad object. For each object, provide a grade and display the results of the isPassed method.
9. Write a Java Program that does the following
10. Create a super class called Car. The Car class has the following fields and methods.

* int speed; double regularPrice; String color; double getSalePrice();

1. Create a sub class of Car class and name it as Truck. The Truck class has the following fields and methods.

* int weight; double getSalePrice();
  + //Ifweight>2000,10% discount. Otherwise,20%discount.

1. Create a subclass of Car class and name it as Ford. The Ford class has the following fields and methods

* int year; intmanufacturerDiscount; double getSalePrice();
  + //From the sale price computed from Carclass, subtract the manufacturer Discount.

1. Create a subclass of Car class and name it as Sedan. The Sedan class has the following fields and methods.

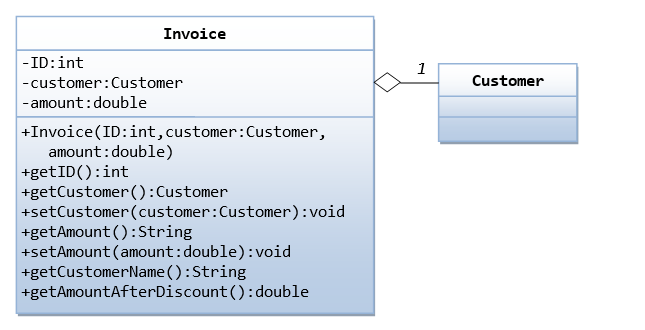
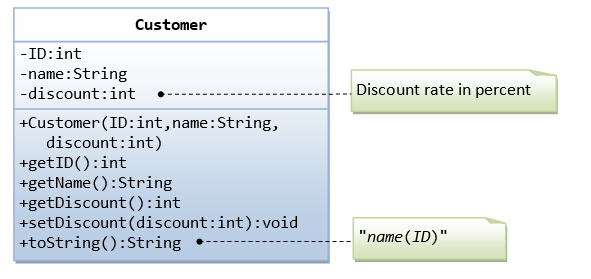
* int length; double getSalePrice();
  + //If length>20feet, 5% discount, Otherwise, 10% discount.

1. Create MyOwnAutoShop class which contains the main() method. Perform the following within the main() method.
   * Create an instance of Sedan class and initialize all the fields with appropriate values.
   * Use super(...) method in the constructor for initializing the fields of the superclass.
   * Create two instances of the Ford class and initialize all the fields with appropriate values
   * Use super(...) method in the constructor for initializing the fields of the super class.
   * Create an instance of Car class and initialize all the fields with appropriate values. Display the sale prices of all instances.
2. Write a Java Program that implements the following

* Define a class **SavingsAccount** with following characteristics.
* Use a static variable **annualInterestRate** to store the annual interest rate for all account holders.
* Private data member **savingsBalance** indicating the amount the saver currently has on deposit.
* Method **calculateMonthlyInterest** to calculate the monthly interest as (savingsBalance \* annualInterestRate / 12). After calculation, the interest should be added to savingsBalance.
* Static method **modifyInterestRate** to set annualInterestRate.
* Parameterized constructor with savingsBalance as an argument to set the value of that instance.
* Test the class SavingsAccount to instantiate two savingsAccount objects, saver1 and saver2, with balances of Rs.2000.00 and Rs3000.00, respectively. Set annualInterestRate to 4%, then calculate the monthly interest and print the new balances for both savers. Then set the annualInterestRate to 5%, calculate the next month’s interest and print the new balances for both savers.

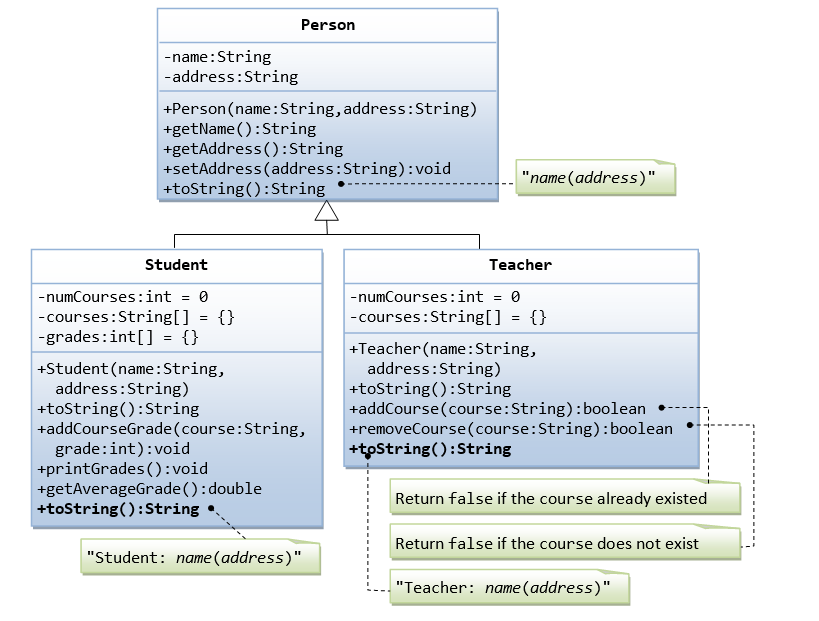
1. Write a Java Program that does the following

* The Customer class models a customer is design as shown in the class diagram. Write the codes for the Customer class and a test driver to test all the public methods.



* The Invoice class, design as shown in the class diagram, composes a Customer instance (written earlier) as its member. Write the codes for the Invoice class and a test driver to test all the public methods.

1. We are required to model students and teachers in our application. We can define a superclass called Person to store common properties such as name and address, and subclasses Student and Teacher for their specific properties. For students, we need to maintain the courses taken and their respective grades; add a course with grade, print all courses taken and the average grade. Assume that a student takes no more than 30 courses for the entire program. For teachers, we need to maintain the courses taught currently, and able to add or remove a course taught. Assume that a teacher teaches not more than 5 courses concurrently.



1. Write a Java Program that implements the following

Create a class Record with following characteristics.

**data members:**

* name[] - to store names of students.
* rank[] - to store ranks of students.

**Member Methods:**

* Record(intnum\_of\_records) – Constructor to initialize member variables with given number of records.
* voidreadvalues(Scanner sc) -  to read names and ranks from a console (e.g. the number of records are 5 then read 5 records).
* void display() - displays the names and the corresponding ranks separated by a single space.

Create a subclass Rank which extends Record class with following characteristics:

**data members:**

* Index: integer to store the index of the topmost rank.

**Member Methods:**

* Rank(intnum\_of\_records) – 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 the index without sorting the array.
* Override toString method to return name and rank of topmost rank student separated by a single space.

1. Implement an abstract class **Reservation** and two subclasses  **ReserveTrain**  and  **ReserveBus**

  Define a **Reservation** abstract class with following characteristics

* Method **reserve** which takes integer value **seats**and **typeOfSeat** as parameters and returns boolean type.
* Method **getAvailableSeats** which return a number of seat remaining.

Define a **ReserveBus** with following characteristics

* Extends the **Reservation** class.
* Data member totalSeats as an integer.
* Parameterized constructor(inttotalSeats) to initialize data member.
* Method **reserve** to reserve the given number of seats and return true and if the required number of seats are not available return false. (**typeOfSeat** parameter has no importance in this method)

Define a **ReserveTrain** with following characteristics

* Extends the **Reservation** class.
* Data members **upperBirthTotalSeats**, **middleBirthTotalSeats** and **lowerBirthTotalSeats** as an integer.
* Parameterized constructor(intlowerBirthTotalSeats, int  middleBirthTotalSeats, intupperBirthTotalSeats) to initialize data members.
* Method **reserve** to reserve the given number of seats for given type of seat and return true and if the required number of seats are not available return false.

1. Implement inheritance using Faculty class

Define a class named **Faculty** with following characteristics

* Parameterized constructor to initialize all data members of a class.
* Data member **name** to store names of the faculty.
* Private data member **basic** of int type to store the basic component of salary.
* **getDetails** method to return name & salary separated by space in "NAME SALARY" format.
* **getSalary** method to calculate and return double type salary.

Define **AssistantProfessor** which extends **Faculty** has following characteristics

* Parameterized constructor to initialize all data members of a class. Call the base class constructor to initialize name & basic.
* Data member **DA** (integer type) which is % of basic salary.
* Override **getSalary** to calculate and return salary as, "basic + ((basic \* DA)/100)".
* **getDetails** method to return name & salary separated by space in "AsstProf NAME SALARY" format.

Define **AssociateProfessor** which extends **AssistantProfessor** has following characteristics

* Parameterized constructor to initialize all data members of a class. Call the base class constructor to initialize other values.
* Data member **MedAllowance** (integer type) which fixed amount.
* Override **getSalary** to calculate and return salary as, basic + ((basic \* DA)/100) + MedAllowance.
* **getDetails** method to return name & salary separated by space in "AsocProf NAME SALARY" format.

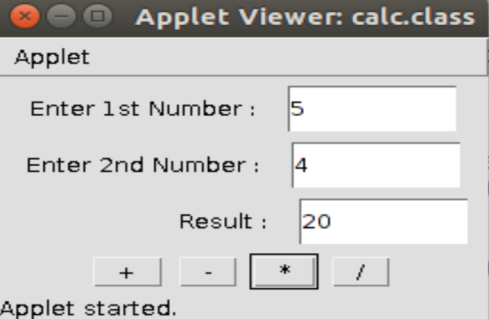
Define **Professor** which extends **AssociateProfessor** has following characteristics

* Parameterized constructor to initialize all data members of a class.
* Data member **OtherAllowance** (integer type) which is % of total income.
* Override **getSalary** to calculate and return salary as, "Salary of AssociateProfessor" + (OtherAllowance% of "Salary of AssociateProfessor").
* **getDetails** method to return name & salary separated by space in "Prof NAME SALARY" format.

1. Write a java program to maintain the student details like USN, Dept names, 3 subject grades and SGPAin student package and keep the staff details such as Staffid, StaffName, designation and subjects handled in a staff package. In main class use these two packages details for Staff and Student classes and display the student and staff information as requested by the user.
2. Write java program to create a package called **AdvMath**, which has two classes. In main class use this package to display the result as requested by the user.
   1. To calculate y = sin(x)+cos(x)+tan(x)
   2. To print pythagorean triplets

**PART-B**

1. Write a program to create two threads t1, t2 which should prints odd numbers, and reverse of a number respectively and stops thread after creating 3 odd numbers.
2. Design an Applet Program to implement the following



1. Write Java program to create an applet with text box. We must type a number in text box first. Then if we press “R” key the reverse of the given number must be displayed on the status bar.
2. Write a program display a "WELCOME" when a mouse button clicked and display "GOOD BYE" when mouse button is released using anonymous inner class.
3. Write a java program to throw a exception (checked) for an employee details• If an employee name is a number, a name exception must be thrown. If an employee age is greater than 50, an age exception must be thrown. Or else an object must be created for the entered employee details.
4. Write Java multithreaded program to implement y=sinx+cosx+tanx
5. Write a Java program to find area of a triangle with three sides a, b, c. A triangle can be formed only if a+b>c, b+c>a, c+a>b. First verify whether the above three conditions are satisfied. If any one of them is not satisfied then throw an exception called ValidateTriangle Exception

Enter the 3 sides of triangle:

7 4 10

Valid Triangle

Enter the 3 sides of triangle:

2 6 8

Not a valid triangle

1. Write a Java program to accept a word and convert it into lowercase if it is in uppercase, and display the new word by replacing only the vowels with the character following it.

For. eg. Example: Sample Input : Computer

Sample Output : cpmpvtfr

1. Write a Java program to display multiplication table of 8 & 9 using shared resources “synchronized displayTable(intnum)”. The table should be displayed with 1 sec delay between every number. First print multiplication table of 8 and then 9.
2. Write a Java program to implement "ADDTION" and "MULTIPLICATION" of two numbers using Lambda Expressions
3. Write a java program to accept a string. Convert the string to uppercase. Count and output the number of double letter sequences that exist in the string.

Sample Input: “SHE WAS FEEDING THE LITTLE RABBIT WITH AN APPLE

Sample Output: 4

1. Write a java program to create five threads with different priorities. Send two threads of highest priority to sleep state. Check the aliveness of the threads.