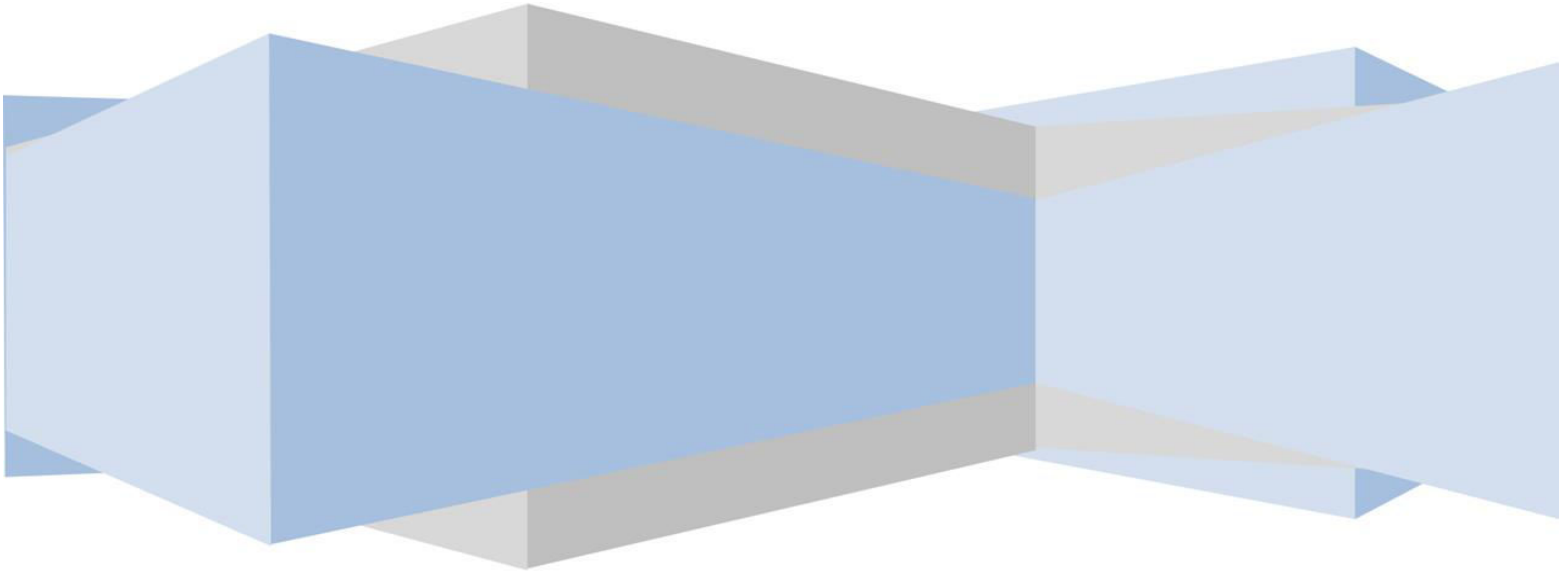
MIET Skill Academy

Java Programming Training Module

Prepared By- Dr.Mukesh Rawat, Mr. Pratik Shrivastava, Mrs. Priyanka Bhardwaj, Ms. Prena Chaudhary, Mr. Vijay Sharma, Mrs. Gunjan Sharma



'*Java*' PROGRAMMING TRAINING MODULE

The Java Fundamentals training module provides the participant to understand Object Oriented Programming concepts and work using Java concepts, datatypes, String class, wrapper class, collection API, exception handing in java. Also teaches about multithreading.

DELIVERY METHOD

25 % Self-paced Learning

75 % Instructor led training

VERSION

Java 8

LEARNING OBJECTIVES

Understanding Object-Oriented Programming Concepts.

- Introduction to Java platform.
- Explain variables, strings, operators and datatype.
- Usage of Java language fundaments using classes and objects.
- Implementation of Object-Oriented Programming Concepts
- Understand and implement Exceptions.
- Understanding and implement multithreading.
- Usage of collection framework.

PREREQUISITES SKILLS

Computer Science fundamentals

- Basic knowledge of any programming and object oriented language, and math.

DURATION

40 Hours

SKILL LEVEL

Basic – Intermediate

HARDWARE REQUIREMENTS

Processor	2 GHz or Higher
GB RAM	8 GB
GB Disk Free	80 GB

SOFTWARE REQUIREMENTS

Operating System	Windows / Linux
Java	Java8 onwards
IDE	Eclipse compactable for Java8

COURSE AGENDA

Lecture 1. Object Oriented Concepts

Duration: 4 Hrs. Overview	This Lecture introduces students to all the components of Object-Orient concepts.
Learning Objectives	After completing this lecture, you should be able to: • Explain what the terms are used in OOP • Difference between Objects and Classes in OOP. • Difference between command line arguments. . Access modifiers in java

Problem 1: Design a class that can be used by a health care professional to keep track of a patient's vital statistics. Here's what the class should do:

1. Construct a class called Patient
2. Store a String name for the patient
3. Store weight and height for patient as doubles
4. Write a method called BMI which returns the patient's BMI as a double. BMI can be calculated as BMI = (Weight in Pounds / (Height in inches x Height in inches)) x 703
5. Next, construct a main method. Create a Patient object and assign some height and weight to that object. Display the BMI of that patient.

Problem 2: Given below a hypothetical table showing rates of Income Tax for male citizens below the age of 65 years.

Taxable income in Rs.	Income tax in Rs.
Does not exceed Rs. 1,60,000	NIL
Greater than 1,60,000 and less than or equal to Rs.5,00,000	(TI-1,60,000)x10%
Greater than 5,00,000 and less than or equal to 8,00,000	[(TI-5,00,000)x20%]+34,000
Is greater than Rs. 8,00,000	[(TI-5,00,000)x30%]+94,000

Create a class **"**IncomeTax**"** with **–**

Member variables-

Age: integer

Gender: string

Taxable_inc: double

Member function **–**

Input()

input the age, gender and taxable income of a person. If the age is more than 65 years or the gender is female, display **"**wrong category**"**.

Calculate_tax()

If the age is less than or equal to 65 years and the gender is male, compute and display the Income Tax Payable as per the table given above.

Problem 3: Define a class library with the following description

Instance variables/ data members

Int acc_num (stores the accession number of book)
String title (stores the title of book)
String author (stores the name of author)
Member methods

Void input() - To input and store the accession number, title and author
Void display () – to display the details in the following format

Accession number Title Author

Lecture 2. Constructors and 1 D Array

Duration: 4 hrs. Overview	This lecture explains about constructors in java and single dimensional arrays.
Learning Objectives	After completing this lecture, you should be able to: • Understand and implement the constructors in Java. • Implement programs on Arrays.

Problem 1: Create a class called **Author** is designed as follows:

It contains: Three private instance variables:

name (String)

email (String)

gender (char of either 'm' or 'f').

One constructor to initialize the name, email and gender with the given values.

Create another class called **Book** is designed as follows:

It contains: Four private instance variables:

bookname (String)

author (of the class Author you have just created)

price (double)

qtyInStock (int)

Assuming that each book is written by one author.

One constructor which constructs an instance with the values given.

Getters and setters:

getBookName(), getAuthor(), getPrice(), setPrice(), getQtyInStock(), setQtyInStock().

There is no setter for name and author.

Print the following (output):

1. Printing the book name, price and qtyInStock from a Book instance. (Hint: aBook.getName())
2. After obtaining the **"**Author**"** object, print the Author (name, email & gender) of the book.

Problem 2 (1D-Array):

Given an array **A** of size **'N'** and an integer **k**, find the maximum for each and every contiguous subarray of size **k**.

Problem 3:

Micro purchased an array *A* having *N* integer values. After playing it for a while, he got bored of it and decided to update value of its element. In one second he can increase value of each array element by *1*. He wants each array element's value to become greater than or equal to *K*. Please help Micro to find out the minimum amount of time it will take, for him to do so.

Lecture 3. Constructors and 1 D Array

Duration: 4 hrs. Overview	This lecture explains about static and this keywords in java and multi dimensional arrays.
Learning Objectives	After completing this lecture, you should be able to: • Understand Single dimensional and multiple dimensional arrays. • Implement programs on static members.

Problem 1:

Reena has 36 packets each containing different no. of candys. She put the packets in 2D pattern. Now she just have to take 7 packets whose candy sum will be maximum. And that 7 packets should be in hourglass form.

Context

Given a *2D Array*, :

```
111000
010000
111000
000000
000000
```

000000

We define an hourglass in to be a subset of values with indices falling in this pattern in 's graphical representation:

```
a b c
  d
e f g
```

There are 16 hourglasses in A, and an HOURGLASS SUM is the sum of an hourglass' values.
Task. Calculate the hourglass sum for every hourglass in A, then print the *maximum* hourglass sum.

Problem 2:
Write a program to check whether the given 2D array of size mxm (where m is odd and m>1) is symmetric along X and Y axis.

Input Format

There are m lines of input, where each line contains m space-separated integers describing *2D Array* A; every value in A will be either 0 or 1.
Explanation:

Consider the 5x5 size 2D array-

```
10101
01010
11111
01010
10101
```

The above 2D array is symmetric along both X and Y axis.

Lecture 4 Implementing OOP Concepts-Inheritance

Duration: 4 hrs. Overview Learning Objectives	This lecture describes about inheritance, After completing this lecture, you should be able to: • Understand and implement inheritance as one of the OOP concept. • Implement programs on single and multilevel inheritance.

Problem-1:
Write a Java program to illustrate the **single inheritance** concept.

Create a class Marks contains the data members **id** of **int** data type, **javaMarks**, **cMarks** and **cppMarks** of **float** data type write a method **setMarks()** to initialize the data members write a method **displayMarks()** which will display the given data

Create another class Result which is derived from the class Marks contains the data members **total** and **avg** of **float** data type write a method **compute()** to find total and average of the given marks write a method **showResult()** which will display the total and avg marks . Write a class SingleInheritanceDemo with **main()** method it receives four arguments as **id**, **javaMarks**, **cMarks** and **cppMarks**.
Create object only to the class Result to access the methods.
If the input given by the user is **"101", "45.50", "67.75", "72.25"** then the program should print the output as:

Id: 101
Java marks: 45.5
C marks: 67.75
Cpp marks: 72.25
Total : 185.5
Avg : 61.833332

Problem-2:
A HighSchool application has two classes: the Person superclass and the Student subclass. Using inheritance, create two new classes, Teacher and CollegeStudent. A Teacher will be like Person but will have additional properties such as salary (the amount the teacher earns) and subject (e.g. **"**Computer Science**"**, **"**Chemistry**"**, **"**English**"**, **"**Other**"**). The CollegeStudent class will extend the Student class by adding a year (current level in college) and major (e.g. **"**Electrical Engineering**"**, **"**Communications**"**, **"**Undeclared**"**). Use this structure to create object of Teacher and CollegeStudent and assign values to their member variables.

Problem-3:
Write a program to create a class named shape. It should contain 2 methods- SetDimension() and Area() which should sets the dimension of the drawing object and calculate its area respectively.
For this class we have three sub classes- Circle, Triangle and Square and each class override the parent class functions- SetDimension () and Area ().
The SetDimension () method should set **"**radius**"**, **"**length and height**"**, **"**length**"** respectively.
The Area() method should calculate **"**Area of Circle**"**, **"**Area of Triangle**"**, **"**Area of Square**"** respectively.
Create objects of Circle, Triangle and Square in the following way and observe the polymorphic nature of the class by calling SetDimension() and Area() method using each object.

Shape c=new Circle();

Shape t=new Triangle();

Shape s=new Square();

Lecture 5. Implementing OOP Concepts-Abstarction

Duration: 4 hrs. Overview	This lecture describes about abstract classes and interfaces.
Learning Objectives	After completing this lectures, you should be able to: • Understand and implement abstraction as one of key concept of OOP in Java. • Implement programs on abstract classes. • Implement programs on Interface and its usage.

Problem 1:

Create an interface called Car with two abstract methods String getName() and int getMaxSpeed(). Also declare one **default** method void applyBreak() which has the code snippet

System.out.println("Applying break on"+ getName());
In the same interface include a **static** method Car getFastestCar(Car car1, Car car2), which returns **car1** if the **maxSpeed** of **car1** is greater than or equal to that of **car2**, else should return **car2**.
Create a class called BMW which implements the interface Car and provides the implementation for the abstract methods **getName()** and **getMaxSpeed()** (make sure to declare the appropriate fields to store **name** and **maxSpeed** and also the constructor to initialize them).

Similarly, create a class called Audi which implements the interface Car and provides the implementation for the abstract methods **getName()** and **getMaxSpeed()** (make sure to declare the appropriate fields to store **name** and **maxSpeed** and also the constructor to initialize them).
Create a **public** class called MainApp with the below code in its **main()** method.

```
public static void main(String args[])
{

    Car car1 = new BMW("BMW X5", 320);
    Car car2 = new BMW("AUDI Q7", 300);
   System.out.println("fastest car is :"+ Car.getFastestCar(car1,car2).getName());

}
```

Problem 2:

Create an abstract class CalcArea and declare the methods **triangleArea(double b, double h)**, **rectangleArea(double l, double b)**, **squareArea(double s)**, **circleArea(double r)**.

Create a class FindArea which extends the abstract class CalcArea used to find areas of triangle,

rectangle, square, circle. Write a class Area with the **main()** method which will receive **two** user inputs and convert them to **double** type. If the input is given as **"1.2","2.7"** then the program should print the output as:
Area of triangle : 1.62
Area of rectangle : 3.24
Area of square : 1.44
Area of circle : 22.8906

Lecture 6 Useful Java API Classes-String class

Duration: 4 hrs. Overview	This lecture explains the usage of Java Application Programming Interfaces (API).
Learning Objectives	After completing this lecture, you should be able to: • Understand the usage of String class. • Implement programs on String class and its functions. And able to perform text processing.

Problem 1:
Write a 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

Problem 2:

Given a string **'S'**, u need to tell whether it is **'sumit's string or not'**.

A string is called 'Sumit's String' , if distance between adjacent character is 1.

Consider that the alphabets are arranged in cyclic manner from **'a' to 'z'**. distance between any character **'x' and 'y'** will be defined as minimum number of steps it takes **'x' to reach 'y'**. Here, character **'x'** can start moving clockwise or anti-clockwise in order to reach at position where character **'y'** is placed. Print **'YES'** if it is Sumit's string else print **'NO'**, for each yest case.

Lecture 7. Useful Java API Classes - StringBuffer

Duration: 8 hrs. Overview	This lecture explains the usage of Java Application Programming Interfaces (API).
Learning Objectives	After completing these lectures, you should be able to: • Understand the usage of growable and writable character sequences. • Implement programs on StringBuffer class and its functions.

Problem 1:

KillCode is trying to learn strings but failing to solve the recursive approach given by his Teacher. His Teacher gave him a string consisting of lower case alphabetes only , He asked to

find a substring in the given string after removing any characters from the original string . For example if the string is **"**Cypher**"** and the substring is "yer", Killcode can remove p,h from the string and can form the given substring.
Now the Teacher increase the difficulty by asking to find the substring and reverse of substring

in given string. If both substring can be formed by a given string by removing certain characters, print **"**GOOD STRING**"** else print **"**BAD STRING**"**.

Problem 2:

Write a program in a class RemovePrefix with a **main** method. The program read one string value. The program should remove the **first two characters** from the argument and print the output, except in one condition. The program should skip removal of **x** or **y** if it encounters them in the first two positions.

Sample Input Output 1
Input: xyz
Output: xyz

Sample Input Output 2
Input: abTreee
Output: Tree

Sample Input Output 3
Input: ayFlower
Output: yFlower

Lecture 8. Use of Wrapper classes

Duration: 4 hrs. Overview	This lecture explain the usage of Java Application Programming Interfaces (API), wrapper class.
Learning Objectives	After completing these lectures, you should be able to: • Implement programs using wrapper class.

Problem 1:
Write a program to input a string and print out the text with the uppercase and lower case letters reversed, but all other characters remain the same as

INPUT: WelComE TO School OUTPUT: wELcOMe to sCHOOL

Problem 2:
Write a program to input a string and extract out the letters, digits and special characters from it.

Problem 3
Find the n-th number whose binary representation is a palindrome
Find the nth number whose binary representation is a palindrome. Do not consider the leading zeros, while considering the binary representation. Consider the 1st number whose binary representation is palindrome as 1, instead of 0

Problem 4:

Next higher number with same number of set bits
Given a number x, find next number with same number of 1 bits in it's binary representation.

For example, consider x = 12, whose binary representation is 1100 (excluding leading zeros on 32 bit machine). It contains two logic 1 bits.
The next higher number with two logic 1 bits is 17 (10001_2).

Lecture 9 Collection Framework - List

Duration: 4 hrs. Overview	This lecture explains array list class of collection framework.
Learning Objectives	After completing this lectures, you should be able to: • Understand the creation of dynamic list using ArrayList and Linked classes. • Implement programs for addition, removing, finding and searching of elements within a list.

Problem 1: Develop a java class with a method saveEvenNumbers(int N) using ArrayList to store even numbers from 2 to N, where N is a integer which is passed as a parameter to the method saveEvenNumbers(). The method should return the ArrayList (A1) created. In the same class create a method printEvenNumbers()which iterates through the arrayList A1 in step 1, and It should multiply each number with 2 and display it in format 4,8,12….2*N. and add these numbers in a new ArrayList (A2).

Problem 2: 1) Create an application for employee management having following classes:
a) Create an Employee class with following attributes and behaviors:
i) EmpIdInt
ii) EmpName String
iii) EmpEmail String
iv) EmpGender char
v) EmpSalaryfloat
vi) GetEmployeeDetails() -> prints employee details
Use an ArrayList which will be used to store the employees and use enumeration/iterator to process the employees.

Problem 3: Create class EmployeeDB which has the following attributes and methods :
i) EmpIdInt
ii) EmpName String
iii) empaddress String
v) EmpSalaryfloat
vi) booleanaddEmployee(Employee e)
vii) booleandeleteEmployee(intempId)
viii) Employee Search(int empid)
Create an ArrayList of Employee(id,name,address,sal) objects and use the above methods.

Lecture 10. Hash table for storage and sorting

Duration: 4 hrs. Overview	This lecture explains creation of hash table and sorting of elements of list.
Learning Objectives	After completing this lecture, you should be able to: • Understand the creation of a collection that uses a hash table for storage. • Implement programs for adding, removing, iterating elements from a Hash Table. • Implement programs using sorted set interface to access the elements in ascending or descending order.

Problem 1:
Create Collection called TreeSet which is capable of storing String objects. The Collection should have the following capabilities
a)Reverse the elements of the Collection
b)Iterate the elements of the TreeSet
c) Checked if a particular element exists or not

Problem 2:
Write a program to store a group of employee names into a HashSet, retrieve the elements one by one using an Iterator.

Problem 3:
Write a program which inputs a text, then implement a code to count the occurrences of each word in a input text and display the word along with corresponding count sorted by the words alphabetically.
(Hint: Use Map as an intermediate collection)

LECTURE-1

Problem-1

Design a class that can be used by a health care professional to keep track of a patient's vital statistics. Here's what the class should do:

1.Construct a class called Patient
2.Store a String name for the patient
3.Store weight and height for patient as doubles
4.Write a method called BMI which returns the patient's BMI as a double. BMI can be calculated as BMI = (Weight in Pounds / (Height in inches x Height in inches)) x 703
5.Next, construct a main method. Create a Patient object and assign some height and weight to that object. Display the BMI of that patient.

Concept:

- Input the value of name, weight and height of patient in the constructor of Patient Class.
- Create a method named BMI and calculate the patient's BMI using the required formula as mentioned in the problem and print name, weight, height and bmi of patient.
- Create another class named Main and call the BMI method in that class.

Solution:

```
import java.util.*;
 class patient
{
   String name;
   double weight;
   double height;
   patient()
   {
      Scanner in = new Scanner(System. in);
      System.out.println("Enter your name: ");
      name = in. nextLine();

       System.out.println("Enter your weight: ");
      weight = in.nextDouble();

      System.out.println("Enter your height: ");
      height = in.nextDouble();
   }

   void BMI()
   {
      double bmi;
      bmi=(weight/(height*height))*703;
      System. out. println("You entered string "+name);
```

```
        System. out. println("Your weight is " + weight);
        System. out. println("Your height is " + height);
        System. out. println("Your bmi is " + bmi);

    }

}
    public class Main
    {
            public static void main(String[] args)
            {
               patient obj1 = new patient();
               obj1.BMI();
            }
    }
```

Problem-2

Given below a hypothetical table showing rates of Income Tax for male citizens below the age of 65 years.

Taxable income in Rs.	Income tax in Rs.
Does not exceed Rs. 1,60,000	NIL
Greater than 1,60,000 and less than or equal to Rs.5,00,000	(TI-1,60,000)x10%
Greater than 5,00,000 and less than or equal to 8,00,000	[(TI-5,00,000)x20%]+34,000
Is greater than Rs. 8,00,000	[(TI-5,00,000)x30%]+94,000

Create a class **"**IncomeTax**"** with –

Member variables-

Age: integer

Gender: string

Taxable_inc: double

Member function –

Input()

input the age, gender and taxable income of a person. If the age is more than 65 years or the gender is female, display **"**wrong category**"**.

Calculate_tax()

If the age is less than or equal to 65 years and the gender is male, compute and display

the Income Tax Payable as per the table given above.

Concept:

- Scan the age and gender in Input() method and check whether person comes in tax category or not.
- Calculate tax as per the information given in table using if-else condition in the method named Calculate_tax().
- Create another class named Main and call Input() and Calculate_tax() methods.

Solution:

```
import java.util.*;

 class IncomeTax
{
   int age;
   String gender;
   double taxable_inc;
   void Input()
   {
       String gender2="female";
      Scanner in = new Scanner(System. in);

      do{
      System.out.println("Enter your age: ");
      age = in.nextInt();
      }
      while(age>65);

      Scanner in1 = new Scanner(System. in);
      do{
      System.out.println("Enter your gender: ");
      gender = in1.nextLine();

      }
      while(gender.equals(gender2));

      System.out.println("Enter your income: ");
      taxable_inc = in.nextDouble();
   }

   void Calculate_tax()
   {
      double tax;
      if (taxable_inc<=160000)
```

```
        {
        System. out. println("tax to be paid is 0 ");
        }
        else if(taxable_inc>160000 && taxable_inc<=500000 )
        {
           tax=(taxable_inc-160000)*0.1;
           System. out. println("tax to be paid is "+tax);
         }
         else if(taxable_inc>500000 && taxable_inc<=800000 )
        {
           tax=((taxable_inc-500000)*0.2)+34000;
           System. out. println("tax to be paid is "+tax);
         }
         else
          {
           tax=((taxable_inc-500000)*0.3)+94000;
           System. out. println("tax to be paid is "+tax);
         }

    }

}
    public class Main
    {
              public static void main(String[] args)
              {
                 IncomeTax obj1 = new IncomeTax();
                 obj1.Input();
                 obj1.Calculate_tax();
              }
    }
```

Problem 3

Define a class library with the following description
Instance variables/ data members
Int acc_num (stores the accession number of book)
String title (stores the title of book)
String author (stores the name of author)
Member methods

Void input() - To input and store the accession number, title and author
Void display () – to display the details in the following format

Accession number Title Author

Approach:

- Create a class library.
- Define variables acc_num,title, author.
- Define input() method for taking input (accession number, title and author) from the user.
- Define display () to display the details in the given format using tab.
- Create another class Main and call input() and display() method in it.

Code:

```
import java.util.*;
class library
{
int acc_num;
String title;
String author;

void input()
{
  Scanner sc=new Scanner(System.in);
  System.out.println("enter accession num");
  acc_num=sc.nextInt();
  sc.nextLine();
  System.out.println("enter titel ");
  title=sc.nextLine();
  System.out.println("enter author ");
  author=sc.nextLine();
}
void display ()
{
 System.out.print("accessing number \t");
 System.out.print("Book title \t");
 System.out.println("Book author");
 System.out.print(acc_num+"\t\t");
 System.out.print( title+"\t\t");
 System.out.println( author);
}
}
public class Main
{
        public static void main(String[] args) {
                library l1=new library();
                l1.input();
                l1.display();
        }
}
```

LECTURE 2

Problem 1: (constructor)

Create a class called **Author** is designed as follows:
It contains:
Three private instance variables:

name (String)
email (String)

gender (char of either 'm' or 'f').

One constructor to initialize the name, email and gender with the given values.

Create another class called **Book** is designed as follows:
It contains:
Four private instance variables:

bookname (String)
author (of the class Author you have just created)
price (double)
qtyInStock (int)

Assuming that each book is written by one author.
One constructor which constructs an instance with the values given.

Getters and setters:
getBookName(), getAuthor(), getPrice(), setPrice(), getQtyInStock(), setQtyInStock().

There is no setter for name and author.

Print the following (output):

3. Printing the book name, price and qtyInStock from a Book instance. (Hint: aBook.getName())
4. After obtaining the **"**Author**"** object, print the Author (name, email & gender) of the book.

Concept:

- Assign the values of name, email and gender in the constructor of class Author
- Pass the values of bookname and Author object in the constructor of class Book
- Design setter and getter methods to set and get price of book, to set and get quantityInStock, to return bookname and to return author name.
- Create another class named Main and call all methods from that class and print the bookname, price and quantityInStock by calling appropriate methods and print Author name, email and gender.

Solution:

```
import java.util.*;
class Author
```

```
{
   String name="";
   String email="";
   char gender;
   Author(String nm,String mail,char g)
   {
      name=nm;
      email=mail;
      gender=g;
   }
}

class Book
{
   String bookname;
   Author auth;
   double price;
   int qtyInStock;
   Book(String bname, Author author)
   {
      bookname=bname;
      auth=author;
   }
   void setPrice(double pr)
   {
      price=pr;
   }
   void setqtyInStock(int n)
   {
      qtyInStock=n;
   }
   double getPrice()
   {
      return price;
   }
   int getqtyInStock()
   {
      return qtyInStock;
   }
   String getName()
   {
      return bookname;
   }
   Author getAuthor()
   {
      return auth;
```

```
    }
}

public class Main
{
        public static void main(String[] args) {
                Scanner sc=new Scanner(System.in);
                String input=sc.nextLine();
                String vals[]=input.split(" ",6);
                Author aob=new Author(vals[0],vals[1],vals[2].charAt(0));
                Book bob=new Book(vals[3],aob);
                double price=Double.parseDouble(vals[4]);

                int qty=Integer.parseInt(vals[5]);
                bob.setPrice(price);
                bob.setqtyInStock(qty);
                Author aob1=bob.getAuthor();

        System.out.println(bob.getName()+"\t"+bob.getPrice()+"\t"+bob.getqtyInStock()+"\t"+ao
b1.name+"\t"+aob1.email+"\t"+aob1.gender);
        }
}
```

Problem-2

Given an array **A** of size **'N'** and an integer **k**, find the maximum for each and every contiguous subarray of size **k**.

Input :

First line contains 2 space separated integers 'N' and 'k'.
Second line contains 'N' space separated integers denoting array elements.

SAMPLE INPUT
9 3
123145236
SAMPLE OUTPUT
3345556

Explanation
First Sub array of size 3 : 1 2 3, here maximum element is 3
second Sub array of size 3 : 2 3 1, here maximum element is 3
third Sub array of size 3 : 3 1 4, here maximum element is 4
fourth Sub array of size 3 : 1 4 5, here maximum element is 5
Fifth Sub array of size 3 : 4 5 2, here maximum element is 5
Sixth Sub array of size 3 : 5 2 3, here maximum element is 5

Seventh Sub array of size 3 : 2 3 6, here maximum element is 6

Concept:

- Take first line input from user using split() method to break it into two parts named N and k.
- Use for-loop to input an array of size N.
- Create method printMax() and pass N and k in it to find maximum element of sub-array.
- In printMax() use nested for-loop. One for-loop to iterate array from beginning to end and another for loop to find maximum element within the range of k elements of array.

Solution:

```
import java.util.*;
public class Main
{
   void printMax(int ar[],int k)
   {
      int max=0;
      for(int i=0;i<ar.length-(k-1);i++)
      {
         max=ar[i];
         for(int j=i;j<(i+k);j++)
         {
            if(ar[j]>max)
            {
               max=ar[j];
            }
         }
         System.out.print(max+"\t");
      }
   }
         public static void main(String[] args) {
                  Scanner sc=new Scanner(System.in);
                  String vals[]=sc.nextLine().split(" ",2);
                  int size=Integer.parseInt(vals[0]);
                  int k=Integer.parseInt(vals[1]);

                  String vals1[]=sc.nextLine().split(" ",size);
                  int arr[]=new int[size];
                  for(int i=0;i<size;i++)
                  {
                     arr[i]=Integer.parseInt(vals1[i]);
                  }
                  Main mob=new Main();
                  mob.printMax(arr,k);
         }
}
```

Problem-3

Micro purchased an array *A* having *N* integer values. After playing it for a while, he got bored of it and decided to update value of its element. In one second he can increase value of each array element by *1*. He wants each array element's value to become greater than or equal to *K*. Please help Micro to find out the minimum amount of time it will take, for him to do so.

Input:

First line consists of two space separated integers denoting *N* and *K*.
Second line consists of *N* space separated integers denoting the array *A*.

SAMPLE INPUT

3 4
1 2 5

SAMPLE OUTPUT

3

Explanation

For first test case,
After *1* second, array will be {2,3,6}
After *2* second, array will be {3,4,7}
After *3* second, array will be {4,5,8}
So it will take *3* second for all array elements to become greater than or equal to *4*.

Concept:

- Take first line of Input from user using split() to break it into two parts namely size and k.
- Use for loop to input an array having elements equal to size.
- Create method printSecond() and pass array and k in it.
- In printSecond() use for loop to find minimum element of array named min.
- Required output will be (k-min).

Solution:

```
import java.util.*;
public class Main{

   void printSecond(int ar[],int k)
   {
      int min=ar[0];
      for(int i=0;i<ar.length;i++)
      {
         if(ar[i]<min)
         {
            min=ar[i];
         }
      }
      System.out.println("seconds required="+(k-min));
   }
   public static void main (String[] args)
   {
```

```
        Scanner sc=new Scanner(System.in);
        String vals[]=sc.nextLine().split(" ",2);
        int size=Integer.parseInt(vals[0]);
        int k=Integer.parseInt(vals[1]);
        String vals1[]=sc.nextLine().split(" ",size);
        int arr[]=new int[size];
        for(int i=0;i<size;i++)
        {
           arr[i]=Integer.parseInt(vals1[i]);
        }
        Main ob=new Main();
        ob.printSecond(arr,k);
    }
}
```

Lecture -3

Problem-1

Reena has 36 packets each containing different no. of candys.She put the packets in 2D pattern.Now she just have to take 7 packets whose candy sum will be maximum. And that 7 packets should be in hourglass form.

Context

Given a *2D Array*, :

```
111000
010000
111000
000000
000000
000000
```

We define an hourglass in to be a subset of values with indices falling in this pattern in 's graphical representation:

```
a b c
  d
e f g
```

There are 16 hourglasses in A, and an HOURGLASS SUM is the sum of an hourglass' values.
Task

Calculate the hourglass sum for every hourglass in A, then print the *maximum* hourglass sum.

Input Format

There are 6 lines of input, where each line contains 6 space-separated integers describing *2D Array* A; every value in A will be in the inclusive range of -9 to 9.

Sample Input

111000
010000
111000
002440
000200
001240 **Sample Output :** 19

Concept:

- Use for-loop to enter values of a matrix row-wise by using split() in String array.
- Use nested for loop to parse above array values into integers and stored values in int array.
- Create method showMax() and pass matrix-array in it. Use nested for-loop to find HOURGLASS sum of matrix.
- Print the maximum HOURGLASS sum as output.

Solution:

```
import java.util.*;
public class Main
{
   int s=0;
   int max=0;
   void showMax(int a[][])
   {
      int c=a[0].length;
      int r=a.length;
      for(int i=0;i<(r-2);i++)
      {
         for(int j=0;j<(c-2);j++)
         {
            s=a[i][j]+a[i][j+1]+a[i][j+2]+a[i+1][j+1]+a[i+2][j]+a[i+2][j+1]+a[i+2][j+2];
            System.out.println(s);
            if(s>max)
               max=s;
         }
       }
         System.out.println("Showing maximum value->"+max);
   }
         public static void main(String[] args) {
```

```
            Scanner sc=new Scanner(System.in);
            String vals[][]=new String[6][6];
            int arr[][]=new int[6][6];
            for(int i=0;i<6;i++)
            {
               vals[i]=sc.nextLine().split(" ",6);
            }
            for(int i=0;i<6;i++)
            {
               for(int j=0;j<6;j++)
               {
                  arr[i][j]=Integer.parseInt(vals[i][j]);
               }
             }
            Main mob=new Main();
            mob.showMax(arr);
         }
}
```

Problem-2

Write a program to check whether the given 2D array of size mxm (where m is odd and m>1) is symmetric along X and Y axis.

Input Format

There are m lines of input, where each line contains m space-separated integers describing *2D Array* A; every value in A will be either 0 or 1.

Explanation:

Consider the 5x5 size 2D array-

```
10101
01010
11111
01010
10101
```

The above 2D array is symmetric along both X and Y axis.

Concept:

- Use nested forloop to take matrix input from user.
- Create SymmetryCheck() to check the symmetry of the matrix-array.
- In SymmetryCheck():-Find mid of rows and columns in variable named xmid and ymid.
- Use nested for loop to check symmetry along y-axis. Outer loop from i=0 to rows and inner loop from j=0 to ymid .
- And Chech if(a[i][j]==a[i][(cols-1)-j]), then matrix is symmetry along Y-axis.
- Apply same logic to check symmetry along x-axis using variable cols and xmid.

Solution:

```
import java.util.*;

public class Main
{
   void SymmetryCheck(int a[][])
   {
      int rows=a.length;
      int cols=a[0].length;
      int ymid=cols/2;
      int xmid=rows/2;
      System.out.println("rows="+rows+"\t cols="+cols);
      System.out.println("ymid="+ymid);

      //y axis symmerty checking

      int flagy=0;
      for(int i=0;i<rows;i++)
      {
         System.out.println("***************");
         for(int j=0;j<ymid;j++)
         {
            System.out.print("j="+j);
            System.out.println(a[i][j]+"\t"+a[i][(cols-1)-j]);
            if(a[i][j]==a[i][(cols-1)-j])
               flagy=0;
            else
               {
               flagy=1;
                System.out.println("Not symetric along Y Axis");
               break;
               }
         }
         if(flagy==1)
            break;
      }
      if(flagy==0)
        System.out.println("Symetric along Y Axis");

      //X axis symmerty checking
      int flagx=0;
      for(int i=0;i<cols;i++)
      {
         System.out.println("***************");
         for(int j=0;j<xmid;j++)
         {
```

```
            System.out.print("j="+j);
            System.out.println(a[j][i]+"\t"+a[(rows-1)-j][i]);
            if(a[j][i]==a[(rows-1)-j][i])
               flagx=0;
            else
               {
               flagx=1;
                System.out.println("Not symetric along X Axis");
               break;
               }
         }
         if(flagx==1)
            break;
      }
      if(flagx==0)
        System.out.println("Symetric along X Axis");
   }
       public static void main(String[] args) {
      Scanner sc=new Scanner(System.in);
      String rc[]=sc.nextLine().split(" ",2);
      int rows=Integer.parseInt(rc[0]);
      int cols=Integer.parseInt(rc[1]);
      String vals[][]=new String[rows][cols];

      for(int i=0;i<rows;i++)
      {
         vals[i]=sc.nextLine().split(" ",cols);
      }
      int arr[][]=new int[rows][cols];
      for(int i=0;i<rows;i++)
      {
         for(int j=0;j<cols;j++)
         {
            arr[i][j]=Integer.parseInt(vals[i][j]);
         }
      }

      Main ob=new Main();
      ob.SymmetryCheck(arr);
         }
}
```

Lecture-4

Problem-1 (Inheritance):

Write a Java program to illustrate the **single inheritance** concept.
Create a class Marks contains the data members
id of **int** data type,
javaMarks, **cMarks** and **cppMarks** of **float** data type
write a method **setMarks()** to initialize the data members
write a method **displayMarks()** which will display the given data
Create another class Result which is derived from the class Marks contains the
data members **total** and **avg** of **float** data type
write a method **compute()** to find total and average of the given marks
write a method **showResult()** which will display the total and avg marks
Write a class SingleInheritanceDemo with **main()** method it receives four
arguments as **id**, **javaMarks**, **cMarks** and **cppMarks**.
Create object only to the class Result to access the methods.
If the input given by the user is **"101", "45.50", "67.75", "72.25"** then the program should print the output as:
Id: 101
Java marks: 45.5
C marks: 67.75
Cpp marks: 72.25
Total : 185.5
Avg : 61.833332

Concept:

1) Create a class Marks,
   - containing variable id(int) , javaMarks(float) , cMarks(float) , cppMarks(float),
   - Write a method setMarks() to initialize the data members.
   - Write a method displayMarks() which will display the given data.
2) Create another class Resulr that extends the Marks class.
   - Add total(float) and avg(float)data members to the class.
   - Write a method showResult() which will display the total and avg marks
3) Write a class SingleInheritanceDemo with main() method it receives four arguments as id, javaMarks, cMarks and cppMarks. Create object only to the class Result to access the methods.

Solution:

```
import java.util.*;
class Marks
{
   int id;
   float javaMarks;
   float cMarks;
   float cppMarks;
```

```
    void setMarks(int id,float javaMarks,float cMarks,float cppMarks)
    {
          this.id=id;
          this.javaMarks=javaMarks;
          this.cMarks=cMarks;
          this.cppMarks=cppMarks;
    }
    void displayMarks()
    {
      System.out.println("id "+id);
      System.out.println("java marks "+javaMarks);
      System.out.println("c marks "+cMarks);
      System.out.println("cppMarks "+ cppMarks);
    }

}
class Result extends Marks
{
   float avg;
   float total;
   void compute()
   {
     total=javaMarks+cppMarks+cMarks;
     avg=total/3;
   }
   void showResult()
   {
      System.out.println("total  "+total);
      System.out.println("average  "+avg);
   }
}

public class Main
{

      public static void main(String[] args) {
                  Scanner sc=new Scanner(System.in);
                  System.out.println("enter user id");
                  int id=sc.nextInt();
                  System.out.println("enter java marks");
                  float java=sc.nextFloat();
                  System.out.println("enter c marks");
                  float c=sc.nextFloat();
                  System.out.println("enter cpp marks");
                  float cpp=sc.nextFloat();
```

```
                Result r1=new Result();

                r1.setMarks(id,java,c,cpp);
                r1.displayMarks();
                r1.compute();
                r1.showResult();
        }
}
```

Problem-2 (Inheritance):

A HighSchool application has two classes: the Person superclass and the Student subclass. Using inheritance, create two new classes, Teacher and CollegeStudent. A Teacher will be like Person but will have additional properties such as salary (the amount the teacher earns) and subject (e.g.**"**Computer Science**"**, **"**Chemistry**"**, **"**English**"**, **"**Other**"**). The CollegeStudent class will extend the Student class by adding a year (current level in college) and major (e.g. **"**Electrical Engineering**"**, **"**Communications**"**, **"**Undeclared**"**). Use this structure to create object of Teacher and CollegeStudent and assign values to their member variables.

Concept

1) Write a Person class.

- Add instance variables to the class for name of type string ,age of type int and gender of type string as Choose appropriate names for the instance variables eg(myName,myAge,myGender).
- Write a constructor for person class to initialize the variables.
- Write "setter" and "getter" methods for all of the class variables.
- Write the toString() method for the Person class.

2) Write a Student class that extends the Person class.

- Add two instance variables for student id and gpa(myIdNum(string),myGPA(double)).
- Write a constructor for Student class to initialize the variables. The constructor will use five parameters to initialize myName, myAge, myGender, myIdNum, and myGPA. Use the super reference to use the constructor in the Person superclass to initialize the inherited values.
- Write "setter" and "getter" methods for all of the class variables.
- Write the toString() method for the Student class. Use a super reference to do the things already done by the superclass.

3) Write a CollegeStudent subclass that extends the Student class.

- Add instance variables to the class for branch and year. Major should be of type String and year of type int.
- Write a constructor for the CollegeStudent class. The constructor will use seven parameters to initialize myName, myAge, myGender, myIdNum, myGPA, year, and branch. Use the super reference to use the constructor in the Student superclass to initialize the inherited values.
- Write "setter" and "getter" methods for all of the class variables. For the CollegeStudent class they would be: getYear, getBranch, setYear, and setBranch.

- Write the toString() method for the CollegeStudent class. Use a super reference to do the things already done by the superclass.

4) Write a Teacher class that extends the parent class Person.

- Add instance variables to the class for subject and salary. Subject should be of type String and salary of type double.
- Write a constructor for the Teacher class. The constructor will use five parameters to initialize myName, myAge, myGender, subject, and salary. Use the super reference to use the constructor in the Person superclass to initialize the inherited values.
- Write "setter" and "getter" methods for all of the class variables. For the Teacher class they would be: getSubject, getSalary, setSubject, and setSalary.
- Write the toString() method for the Teacher class. Use a super reference to do the things already done by the superclass

5) Write a testing class with a main() .

Solution:

```
import java.util.*;
class Person{
 private String myName ;
 private int myAge;
 private String myGender;

 public Person(String name, int age, String gender){
 myName = name;
 myAge = age;
 myGender = gender;
 }

 public String getName(){
 return myName;
 }

public int getAge(){
 return myAge;
}
 public String getGender(){
 return myGender;
 }
 public void setName(String name){
 myName = name;
 }
 public void setAge(int age){
 myAge = age;
 }
```

```
 public void setGender(String gender){
 myGender = gender;
 }
 public String toString(){
 return myName + ", age: " + myAge + ", gender: " + myGender;
 }
}

class Student extends Person
{
private String myIdNum;
private double myGPA;

public Student(String name, int age, String gender,String idNum, double gpa)
{
   super(name, age, gender);
   myIdNum = idNum;
   myGPA = gpa;
}
public String getIdNum(){
return myIdNum;
}
public double getGPA(){
return myGPA;
 }
public void setIdNum(String idNum){
myIdNum = idNum;
}
public void setGPA(double gpa){
myGPA = gpa;
}
// overrides the toString method in the parent class
public String toString(){
return super.toString() + ", student id: " + myIdNum + ", gpa: " + myGPA;
}
}
class collegeStudent extends Student
{
   private int year;
   private String branch;

   public collegeStudent(String name, int age, String gender,String idNum, double gpa,int
year,String branch){
super(name,age,gender,idNum,gpa);
this.year = year;
this.branch =branch;
```

```
}
   public void setYear(int year)
   {
      this.year=year;
   }
   public void setBranch(String branch)
   {
      this.branch=branch;
   }
   public int getYear()
   {
      return year;
   }
   public String getBranch()
   {
      return branch;
   }
 public String toString(){
return super.toString() + ", year: " + year + ", branch: " + branch;
}
}

class Teacher extends Person
{
   private float salary;
   private String subject;

   public Teacher(String name, int age, String gender,float salary, String subject){
// use the super class' constructor
super(name, age, gender);
// initialize what's new to Student
this.salary = salary;
this.subject = subject;
}
   public void setSalary(float salary )
   {
      this.salary=salary;
   }
   public void setSubject(String subject)
   {
      this.subject=subject;
   }
   public float getSalary()
   {
      return salary ;
   }
```

```
    public String getSubject()
    {
       return subject;
    }
    public String toString(){
return super.toString() + ", salary: " + salary + ", subject: " + subject;
}
}

public class Main
{
        public static void main(String[] args) {

    Person a = new Person("Anil",23,"male");
    Student b = new Student("jimmy", 32,"male","123" ,1);

    collegeStudent cg1=new collegeStudent("priya",23,"f","1",9,2020,"cse");
    Teacher c = new Teacher("Mike",23,"m", 95000,"maths");
    System.out.println(a);
    System.out.println(b);
    System.out.println(cg1);
        System.out.println(c);

        }
}
```

Problem-3 (Method overridding):

Write a program to create a class named shape. It should contain 2 methods- SetDimension() and Area() which should sets the dimension of the drawing object and calculate its area respectively.

For this class we have three sub classes- Circle, Triangle and Square and each class override the parentclass functions- SetDimension () and Area ().

The SetDimension () method should set **"**radius**"**, **"**length and height**"**, **"**length**"** respectively.

The Area() method should calculate **"**Area of Circle**"**, **"**Area of Triangle**"**, **"**Area of Square**"** respectively.

Create objects of Circle, Triangle and Square in the following way and observe the polymorphic nature of the class by calling SetDimension() and Area() method using each object.

Shape c=new Circle();

Shape t=new Triangle();

Shape s=new Square();

Concept:

1) Create a class shape. Add two methods - SetDimension() and Area().
2) Create a class Circle. Give the definition of take input for Circle in setd() and calculate and print
area of circle in area().
3) Create a class Triangle. Give the definition of take input for Triangle in setd() and calculate and
print area of circle in area().
4) Create a class Sqaure. Give the definition of take input for Square in setd() and calculate and print
area of circle in area().
5) Create main class
In main class Create objects of Circle, Triangle and Square in the following way
Shape c=new Circle();
Shape t=new Triangle();
Shape s=new Square();

Solution:

```
import java.lang.*;
import java.io.*;

 class Shape
 {
   void setd()
   {
      System.out.println("enter dimensions");
   }
   area()
   {
     System.out.println("area of figure");
   }

}

class Square extends Shape {

 void setd() {
 double side;
Scanner sc=new Scanner(System.in);
System.out.println("enter side dimension");
side=sc.nextDouble();
 }

 void area() {
```

```
  int sqaureArea=side*side;
  System.out.println ("Area of Rectangle = "+sqaureArea);
 }
}

class Triangle extends Shape
{
 void setd() {
  double base;
  double height;
    Scanner sc=new Scanner(System.in);
    System.out.println("enter base");
    base=sc.nextDouble();
    System.out.println("enter height");
    height=sc.nextDouble();
  }

 void area() {

  double triangleArea=((base*height)/2);
  System.out.println ("Area of Triangle = "+triangleArea);
 }
}
class Circle extends Shape
{
 void setd() {
   double radius;
   Scanner sc=new Scanner(System.in);
   System.out.println("enter radius of circle");
   }

 void area() {
  double pi = 3.14;
  double circleArea=p1*r*r;
  System.out.println ("Area of Triangle = "+circleArea);
 }
}

public class Main
{
public static void main(String[] args) {
     Shape S1= new Square();
     S1.setd();
     S1.area();
```

```
    Shape S2=  new Triangle();
    S2.setd();
    S2.area();

    Shape S3 =new Circle();
    S3.setd();
    S3.area();

        }

}
```

Lecture-5

Problem 1 (Interfaces):

Create an interface called Car with two abstract methods String getName() and int getMaxSpeed().
Also declare one **default** method void applyBreak() which has the code snippet
System.out.println(**"**Applying break on**"**+ getName());
In the same interface include a **static** method Car getFastestCar(Car car1, Car car2),
which returns **car1** if the **maxSpeed** of **car1** is greater than or equal to that of **car2**, else should return **car2**.
Create a class called BMW which implements the interface Car and provides the implementation for the abstract methods **getName()** and **getMaxSpeed()** (make sure to declare the appropriate fields to store **name** and **maxSpeed** and also the constructor to initialize them).
Similarly, create a class called Audi which implements the interface Car and provides the implementation for the abstract methods **getName()** and **getMaxSpeed()** (make sure to declare the appropriate fields to store **name** and **maxSpeed** and also the constructor to initialize them).
Create a **public** class called MainApp with the below code in its **main()** method.

```
public static void main(String args[])
{
Car car1 = new BMW("BMW X5", 320);
Car car2 = new BMW("AUDI Q7", 300);
System.out.println("fastest car is :"+ Car.getFastestCar(car1,car2).getName());
}
```

Concept:

1. Create an interface named Car
    a. In this, create methods getName(), getMaxSpeed(), applyBreak() and getfastestCar().
    b. In getFastestCar() method, apply the logic to find the fastest car as given in problem statement.
2. Create 2 sub classes named BMW and AUDI
    a. Create constructor to initialize the values
    b. For above two sub classes, define the body of getName(), getMaxSpeed() and applyBreak() methods.

3. Create a main class and object and obtain the name for the fastest car.

Solution:

```
interface Car {
	public String getName();
	public int getMaxSpeed();
	public void applyBreak();
	public static Car getFastestCar(Car car1, Car car2)
	{
		if(car1.getMaxSpeed() >= car2.getMaxSpeed())
		{
			return car1;
		}
		else
		{
			return car2;
		}
	}
}

class BMW implements Car {
	String cname;
	int bmwspeed;
	BMW(String cn, int speed)
	{
		cname=cn;
		bmwspeed=speed;
	}
	public String getName()
	{
		return cname;

	}
	public int getMaxSpeed()
	{
		return bmwspeed;
	}
	public void applyBreak()
	{
		System.out.println("Applying break on "+getName());
	}


}

class Audi implements Car {
```

```
        String cname;
        int audispeed;
        Audi(String cn, int speed)
        {
                cname=cn;
                audispeed=speed;
        }
        public String getName()
        {
                return cname;
        }
        public int getMaxSpeed()
        {
                return audispeed;
        }
        public void applyBreak()
        {
                System.out.println("Applying break on "+getName());
        }

}

public class MainApp {
        public static void main(String args[]) {

        Car car1=new BMW("BMW X5", 320);
        Car car2=new BMW("AUDI Q7", 300);

        System.out.println("Fastest car is : " + Car.getFastestCar(car1,car2).getName());
        }
}
```

Problem 2(Abstract class):

Create an abstract class CalcArea and declare the methods **triangleArea(double b, double h)**, **rectangleArea(double l, double b)**, **squareArea(double s)**, **circleArea(double r)**.

Create a class FindArea which extends the abstract class CalcArea used to find areas of triangle, rectangle, square, circle. Write a class Area with the **main()** method which will receive **two** user inputs and convert them to **double** type. If the input is given as **"1.2","2.7"** then the program should print the output as:

Area of triangle : 1.62
Area of rectangle : 3.24
Area of square : 1.44
Area of circle : 22.8906

Concept:

1.Create an abstract class calcArea

   a. Create findTriangle(), findRectangle(),findSquare(),findCircle() methods as abstract.

2. Create a sub class named findArea which extends the calcArea.
3. Define the body for all the abstract methods to achieve the area of respective shape.
4. Create the main class and object and call the defined methods through that object.

Solution:

```
abstract class calcArea {
   abstract void findTriangle(double b, double h);
   abstract void findRectangle(double l, double b);
   abstract void findSquare(double s);
   abstract void findCircle(double r);
}
 class findArea extends calcArea {

   void findTriangle(double b, double h)
   {
      double area = (b*h)/2;
      System.out.println("Area of Triangle: "+area);
   }
    void findRectangle(double l, double b)
   {
      double area = l*b;
      System.out.println("Area of Rectangle: "+area);
   }
    void findSquare(double s)
   {
      double area = s*s;
      System.out.println("Area of Square: "+area);
   }
    void findCircle(double r)
   {
      double area = 3.14*r*r;
      System.out.println("Area of Circle: "+area);
   }
}

class area {
   public static void main(String args[])
   {
      double l, b, h, r, s;
```

```
        findArea area = new findArea();
        Scanner get = new Scanner(System.in);

        System.out.print("\nEnter Base & Vertical Height of Triangle: ");
        b = get.nextDouble();
        h = get.nextDouble();
        area.findTriangle(b, h);

        System.out.print("\nEnter Length & Breadth of Rectangle: ");
        l = get.nextDouble();
        b = get.nextDouble();
        area.findRectangle(l, b);

        System.out.print("\nEnter Side of a Square: ");
        s = get.nextDouble();
        area.findSquare(s);

        System.out.print("\nEnter Radius of Circle: ");
        r = get.nextDouble();
        area.findCircle(r);
    }
}
```

Lecture-6

Problem 1 (String handling):

Write a 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

Concept:

1. Input a string from user and convert it to uppercase.
2. Iterate all the characters of the input string and matches
   Character(i) with Character(i+1)
   If both the character matches then increment the counter variable by 1

Solution:

```
import java.util.*;
public class Main
{
   void doubleOccurence(String str)
   {
      String s1=str.toUpperCase();
      System.out.println(s1);
      int count=0;
      for(int i=0;i<s1.length()-1;i++)
      {
         if(s1.charAt(i)==s1.charAt(i+1))
         {
            count++;
         }
      }
      System.out.println(count);

   }
         public static void main(String[] args) {
            Scanner sc=new Scanner(System.in);
            String str=sc.nextLine();
            Main ob=new Main();
            ob.doubleOccurence(str);

         }
}
```

Problem 2 (String handling):

Given a string **'S'**, u need to tell whether it is **'sumit's string or not'**.

A string is called 'Sumit's String' , if distance between adjacent character is 1.

Consider that the alphabets are arranged in cyclic manner from **'a' to 'z'**. distance between any character **'x' and 'y'** will be defined as minimum number of steps it takes **'x' to reach 'y'**. Here, character **'x'** can start moving clockwise or anti-clockwise in order to reach at position where character **'y'** is placed. Print **'YES'** if it is Sumit's string else print **'NO'**, for each test case.

Concept:

1. Input the string from user.
2. Iterate all the characters of the input string and check
   Character(i)- Character(i+1)=1 or -1
   Then print “yes” else “No”

Solution:

```
import java.util.*;
public class Main
{
   void sumitstring(String str[], int count)
   {
      int fg=0,res_count=0;
      String res[]=new String[count];
      for(int k=0;k<str.length;k++)
      {
         for(int i=0;i<str[k].length()-1;i++)
         {
         if(((int)str[k].charAt(i)-(int)str[k].charAt(i+1))==1 || ((int)str[k].charAt(i)-
(int)str[k].charAt(i+1))==-1)
            {
               fg=0;
            }
         else
            {
               fg=1;
               res[res_count]="NO";
               res_count++;
               break;
            }
         }
         //System.out.println("Yes");
         if(fg==0)
         {
```

```
            res[res_count]="YES";
            res_count++;
            }
        }
        for(int i=0;i<res_count;i++)
        {
            System.out.println(res[i]);
        }
    }
        public static void main(String[] args) {
                Scanner sc=new Scanner(System.in);
                int tcount=Integer.parseInt(sc.nextLine());
                String arr[]=new String[tcount];
                for(int c=0;c<tcount;c++)
                {
                   arr[c]=sc.nextLine();
                }
                Main ob=new Main();
                ob.sumitstring(arr,tcount);
        }
}
```

Lecture-7

Problem 1:

KillCode is trying to learn strings but failing to solve the recursive approach given by his Teacher. His Teacher gave him a string consisting of lower case alphabetes only , He asked to

find a substring in the given string after removing any characters from the original string . For example if the string is **"**Cypher**"** and the substring is "yer", Killcode can remove p,h from the string and can form the given substring.

Now the Teacher increase the difficulty by asking to find the substring and reverse of substring in given string. If both substring can be formed by a given string by removing certain characters, print **"**GOOD STRING**"** else print **"**BAD STRING**"**.

SAMPLE INPUT
abta
aba
oley
le
tereo
re

SAMPLE OUTPUT
GOOD STRING
BAD STRING
GOOD STRING

Concept:
1.Input a text string-str and substring-sstr
2.Iterate all the characters of the substring and check
Condition-1:

Indexof(character(i)) in str < Indexof(character(i+1)) in str
Then,
Reverse substring
Iterate all the characters of the substring and check
Condition 2:

Indexof(character(i)) in str < Indexof(character(i+1)) in str
3. If condition-1 and condition-2 satisfies then,
Print “Good String”
Else
Print “Bad String”

Solution:

```
import java.util.*;
public class Main
{
      int x1=0,x2=-1;
      int f;
      int exc_count=0;
   void check(String str,String sstr)
   {
      System.out.println(str+"\t"+sstr);
      x1=0;
      x2=-1;
      f=0;
      for(int i=0;i<sstr.length();i++)
      {
         x1=str.indexOf(sstr.charAt(i),x2+1);
         System.out.println("x1="+x1);
         if(x1==-1)
         {
            f=1;
            break;
         }
         else
```

```
        {
            f=0;
            x2=x1;
        }
    }
        System.out.println("value of f="+f);
    if(f==1)
    {
        System.out.println("value of f="+f);
        System.out.println("Bad String");
    }
    else
    {
        StringBuffer sb=new StringBuffer(sstr);
        checkReverse(str,sb.reverse().toString());
    }
}

  void checkReverse(String str,String sstr)
  {
    System.out.println(str+"\t"+sstr);
    x1=0;
    x2=-1;
    f=0;
    for(int i=0;i<sstr.length();i++)
    {
        x1=str.indexOf(sstr.charAt(i),x2+1);
        System.out.println("x1="+x1);
        if(x1==-1)
        {
            f=1;
            break;
        }
        else
        {
            f=0;
            x2=x1;
        }
    }
        //System.out.println("value of f="+f);
    if(f==1)
    {
        System.out.println("value of f="+f);
        System.out.println("Bad String");
    }
    else
```

```
        {
            System.out.println("value of f="+f);
            System.out.println("Good String");
        }
    }
        public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String str=s.nextLine();
        String sstr=s.nextLine();
        Main ob=new Main();
                    ob.check(str,sstr);
    }
}
```

Problem 2:

Write a program in a class RemovePrefix with a **main** method. The program read one string value. The program should remove the **first two characters** from the argument and print the output, except in one condition. The program should skip removal of **x** or **y** if it encounters them in the first two positions.

Sample Input Output 1
Input: xyz
Output: xyz

Sample Input Output 2
Input: abTreee
Output: Tree

Sample Input Output 3
Input: ayFlower
Output: yFlower

Concept:
1.Input the text string-str
2.Extract the starting 2 characters of str
3.Check the following conditions-
  Condition-1-
    Both characters=”xy”
  Condition-2-
    first character=”x” or “y”
  condition-3-
    second character = “x” or “y”
  condition -4
    both characters are different
4. If condition-1 and 2 satisfies then print str as it is.
  If condition – 3 satisfies then, remove first character and print rest of string

If condition-4 satisfies then, remove first 2 characters and print rest of string

Solution:

```
import java.util.*;

public class Main
{
   void show(String str)
   {
      System.out.println(str);
      if(str.startsWith("xy")||str.startsWith("x")||str.startsWith("y"))
      {
         System.out.println(str);
      }
      else
      {
         if(str.charAt(1)=='x' || str.charAt(1)=='y')
         {
            System.out.println(str.substring(1));
         }
         else
         {
            System.out.println(str.substring(2));
         }

      }
   }
         public static void main(String[] args) {
                  Scanner sc=new Scanner(System.in);
                  String s=sc.nextLine();
                  Main ob=new Main();
                  ob.show(s);
         }
}
```

Lecture – 8

Problem 1:
Write a program to input a string and print out the text with the uppercase and lower case letters reversed, but all other characters remain the same as

INPUT: WelComE TO School OUTPUT: wELcOMe to sCHOOL

Concept:

1. Take one string of any length and calculate its length.
2. Scan string character by character and keep checking the index .
   - If character in a index is in lower case, then subtract 32 to convert it in upper case, else add 32 to convert it in upper case
3. Print the final string.

Solution:

```
import java.io.*;
public class question1
{
   public static void main(String args[]) throws IOException
   {
      int ch;
      BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
      System.out.println("ENTER THE STRING ");
      String input=br.readLine();
      int len=input.length();
      int i;
      for(i=0; i<len; i++)
      {
         ch=input.charAt(i);
         if(ch>=65 && ch<=90)
         {
            ch=ch+32;
            System.out.print((char)ch);
         }
         else if(ch>=97 && ch<=122)
         {
            ch=ch-32;
            System.out.print((char)ch);
         }
         else
         {
            System.out.print((char)ch);
         }
      }
   }}
```

Problem 2:

Write a program to input a string and extract out the letters, digits and special characters from it.
Sample input:
As123df@#

Sample Output:
Letters : Asdf
Digits: 123
Special characters : @#

Concept:

1. Calculate the length of the string.
2. Scan each every character(ch) of a string one by one
   - if (ch is a digit) then append it in res1 string.
   - else if (ch is alphabet) append in string res2.
   - else append in string res3.
3. Print the all the strings, we will have one string containing numeric part, other non numeric part and last one contain special characters.

Solution:

```
import java.io.*;
class GFG {
	static void charCheck(char input_char)
	{
		// CHECKING FOR ALPHABET
		if ((input_char >= 65 && input_char <= 90)
			|| (input_char >= 97 && input_char <= 122))
			System.out.println(" Alphabet ");

		// CHECKING FOR DIGITS
		else if (input_char >= 48 && input_char <= 57)
			System.out.println(" Digit ");

		// OTHERWISE SPECIAL CHARACTER
		else
			System.out.println(" Special Character ");
	}
	public static void main(String[] args)
	{
		char input_char = '$';
		charCheck(input_char);
	}
}
```

Problem 3
Find the n-th number whose binary representation is a palindrome
Find the nth number whose binary representation is a palindrome. Do not consider the leading zeros, while considering the binary representation. Consider the 1st number whose binary representation is palindrome as 1, instead of 0

Examples:
Input:1
Output:1
1st number whose binary representation is palindrome is 1 (1)

Output: 27
9th number whose binary representation is palindrome is 27 (11011)

Concept:
1) We can divide the set of palindrome numbers into some groups.
2) n-th group will have (2^(n-1) + 2^n = 3 * 2 ^(n-1)) number of binary palindromes
3) With the given number, we can find the group to which it belongs to and the offset in that group.
4) As the leading zeros are not to be considered, we should use bit 1 as the starting bit and ending bit of the number in binary representation
5) And we will fill other bits based on the groupno and groupoffset
6) Based on the offset, we can find which bit should be inserted at the middle (|(nothing) or 0 or 1 and which number(in binary form) (1 or 2 or 3 or 4 or ..) should be placed in both directions from middle.

Solution:
```
import java.io.*;
class GFG
{
static int INT_MAX = 2147483647;

/* Finds if the kth bit
is set in the binary
representation */
static int isKthBitSet(int x, int k)
{
        return ((x & (1 <<
                        (k - 1))) > 0) ? 1 : 0;
}

/* Returns the position of
leftmost set bit in the
binary representation */
static int leftmostSetBit(int x)
{
```

```
        int count = 0;
        while (x > 0)
        {
                count++;
                x = x >> 1;
        }
        return count;
}

/* Finds whether the integer
in binary representation is
palindrome or not*/
static int isBinPalindrome(int x)
{
        int l = leftmostSetBit(x);
        int r = 1;

        // One by one compare bits
        while (l > r)
        {

                // Compare left and right
                // bits and converge
                if (isKthBitSet(x, l) !=
                        isKthBitSet(x, r))
                        return 0;
                l--;
                r++;
        }
        return 1;
}

static int findNthPalindrome(int n)
{
        int pal_count = 0;

        /* Start from 1, traverse
        through all the integers */
        int i = 0;
        for (i = 1; i <= INT_MAX; i++)
        {
                if (isBinPalindrome(i) > 0)
                {
                        pal_count++;
                }
```

```
                /* If we reach n,
                break the loop */
                if (pal_count == n)
                        break;
        }

        return i;
}
// Driver code
public static void main (String[] args)
{
        int n = 9;
        System.out.println(findNthPalindrome(n));
}
}
```

Problem 4:

Next higher number with same number of set bits

Given a number x, find next number with same number of 1 bits in it's binary representation.

For example, consider x = 12, whose binary representation is 1100 (excluding leading zeros on 32 bit machine). It contains two logic 1 bits.

The next higher number with two logic 1 bits is 17 (10001_2).

Concept:

When we observe the binary sequence from 0 to $2^n - 1$ (n is # of bits), right most bits (least significant) vary rapidly than left most bits. The idea is to find right most string of 1's in x, and shift the pattern to right extreme, except the left most bit in the pattern. Shift the left most bit in the pattern (omitted bit) to left part of x by one position. An example makes it more clear,

x_{10}= 156

x_2 = 10011100

10011100

00011100 - right most string of 1's in x

00000011 - right shifted pattern except left most bit ------> [A]

00010000 - isolated left most bit of right most 1's pattern

00100000 - shiftleft-ed the isolated bit by one position ------> [B]

10000000 - left part of x, excluding right most 1's pattern ------> [C]

10100000 - add B and C (OR operation) ------> [D]

10100011 - add A and D which is required number 163

Solution:

```
class GFG
{

// this function returns next higher
// number with same number of set bits as x.
static int snoob(int x)
{

int rightOne, nextHigherOneBit, rightOnesPattern, next = 0;

if(x > 0)
{

   // right most set bit
   rightOne = x & -x;

   // reset the pattern and set next higher bit
   // left part of x will be here
   nextHigherOneBit = x + rightOne;

   // nextHigherOneBit is now part [D] of the above explanation.

   // isolate the pattern
   rightOnesPattern = x ^ nextHigherOneBit;

   // right adjust pattern
   rightOnesPattern = (rightOnesPattern)/rightOne;

   // correction factor
   rightOnesPattern >>= 2;

   // rightOnesPattern is now part [A] of the above explanation.

   // integrate new pattern (Add [D] and [A])
   next = nextHigherOneBit | rightOnesPattern;
}
 return next;
}
 public static void main (String[] args)
{
   int x = 156;
   System.out.println("Next higher number with same" +
                "number of set bits is "+snoob(x));
}
}
```

Lecture - 9

Problem 1:

Develop a java class with a method saveEvenNumbers(int N) using ArrayList to store even numbers from 2 to N, where N is a integer which is passed as a parameter to the method saveEvenNumbers(). The method should return the ArrayList (A1) created. In the same class create a method printEvenNumbers()which iterates through the arrayList A1 in step 1, and It should multiply each number with 2 and display it in format 4,8,12….2*N. and add these numbers in a new ArrayList (A2).

Concept:
Delare an array of ArrayList object.
Add even numbers in the even location of the ArrayList.
Delare an array ArrayList object.
Add mutlplication result of 1st ArrayList value and 2 in an another array of ArrayList.
Display such numbers in the screen.

Solution:

```
import java.util.ArrayList;
public class Assignment1 {

	private ArrayList<Integer> list = new ArrayList<Integer>();
	public ArrayList<Integer> saveEvenNumbers(int N) {
		list = new ArrayList<Integer>();
		for (int i = 2; i <= N; i++) {
			if (i % 2 == 0) list.add(i);
		}
		return list;
	}
	public ArrayList<Integer> printEvenNumbers() {
		ArrayList<Integer> newList = new ArrayList<Integer>();
		for (int item : list) {
			newList.add(item * 2);
		System.out.println(item * 2);
		}
		return newList;
}
	public static void main(String[] args) {
	Assignment1 asg = new Assignment1();
	asg.saveEvenNumbers(10);
	asg.printEvenNumbers();
	}
}
```

Problem 2:

1) Create an application for employee management having following classes:
a) Create an Employee class with following attributes and behaviors:
i) EmpIdInt
ii) EmpName String
iii) EmpEmail String
iv) EmpGender char
v) EmpSalaryfloat
vi) GetEmployeeDetails() -> prints employee details
Use an ArrayList which will be used to store the employees and use enumeration/iterator to process the employees.

Concept:
Delare Class of employee.
Declare an VectorList object.
Add object of employee class in VectorList.
Use inbuilt iterator object to iterate in the VectorList.

Solution:

```
import java.util.Iterator;
import java.util.Vector;
class Employee {
	private int id;
	private String name;
	private String address;
	private Double salary;
	public Employee(int id, String name, String address, Double salary) {
		super();
		this.id = id;
		this.name = name;
		this.address = address;
		this.salary = salary;
	}
	public int getId() {
		return id;
	}
	@Override
	public String toString() {

		return "Employee [id=" + id + ", name=" + name + ", address=" + address + ",
salary=" + salary + "]";
	}
}
public class MainClass{
	public static void main(String[] args) {
```

```
                Vector<Employee> list = new Vector<>();
                list.add(new Employee(101, "Bob", "123 street, India", 20000.0));
                list.add(new Employee(102, "Alice", "234 street, India", 30000.0));
                list.add(new Employee(103, "John", "345 street, India", 25000.0));
                list.add(new Employee(104, "Stuart", "456 street, India", 40000.0));
                Iterator<Employee> it = list.iterator();
                while (it.hasNext())
                System.out.println(it.next());
        }

}
```

Problem 3:

Create class EmployeeDB which has the following attributes and methods :
i) EmpIdInt
ii) EmpName String
iii) empaddress String
v) EmpSalaryfloat
vi) booleanaddEmployee(Employee e)
vii) booleandeleteEmployee(intempId)
viii) Employee Search(int empid)
Create an ArrayList of Employee(id,name,address,sal) objects and use the above methods.

Concept:

Delare Class of employee.
Declare an List object.
Add object of employee class in List.
Use inbuilt iterator object & iteration method to iterate in the VectorList.

Solution:

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
class Employee {
        private int id;
        private String name;
        private String address;
        private Double salary;
        public Employee(int id, String name, String address, Double salary) {
                super();
                this.id = id;
                this.name = name;
                this.address = address;
                this.salary = salary;
```

```
	}
	public int getId() {
		return id;
	}
	public String toString() {
		return "Employee [id=" + id + ", name=" + name + ", address=" + address + ",
salary=" + salary + "]";
	}
}
public class Assignment5 {
	public static void main(String[] args) {
		List<Employee> list = new ArrayList<>();
		list.add(new Employee(101, "Bob", "123 street, India", 20000.0));
		list.add(new Employee(102, "Alice", "234 street, India", 30000.0));
		list.add(new Employee(103, "John", "345 street, India", 25000.0));
		list.add(new Employee(104, "Stuart", "456 street, India", 40000.0));
		Iterator<Employee> it = list.iterator();
		int searchId = 102;
		while (it.hasNext()) {
			Employee emp = it.next();
			if (emp.getId() == searchId)
			System.out.println(emp);
		}
	}
}
```

Lecture 10

Problem 1:

Create Collection called TreeSet which is capable of storing String objects. The Collection should have the following capabilities
a)Reverse the elements of the Collection
b)Iterate the elements of the TreeSet
c) Checked if a particular element exists or not

Concept:
Delare Class of Assignment3.
Declare an TreeSet object.
Add object of Assignment3 class in TreeSet.
Use inbuilt iterator object & iteration method to iterate in the TreeSet.

Solution:

```
import java.util.Iterator;
import java.util.TreeSet;
public class Assignment3 {
	public static void main(String[] args) {
		TreeSet<String> set = new TreeSet<>();
		//Collection<String> set = new TreeSet<>(Collections.reverseOrder());
		set.add("Bob");
		set.add("Alice");
		set.add("John");
		set.add("Richard");
		Iterator<String> it = set.iterator();
		String query = "John";
		boolean result = false;
		while (it.hasNext()) {
			if (it.next().equals(query)) {
			result = true;
			break;
			}
		}
		if (result) System.out.println(query + " exists");
		else System.out.println(query + " doesn't exist");
	}
}
```

Problem 2:

Write a program to store a group of employee names into a HashSet, retrieve the elements one by one using an Iterator.

Concept:

Delare Class of Assignment2.
Declare an HashSet object.
Add object of Assignment2 class in HashSet.
Use inbuilt iterator object & iteration method to iterate in the HashSet.

Solution:

```
import java.util.HashSet;
import java.util.Iterator;
public class Assignment2 {
	public static void main(String[] args) {
		HashSet<String> set = new HashSet<>();
		set.add("Bob");
		set.add("Alice");
		set.add("John");
	set.add("Richard");
		Iterator<String> it = set.iterator();
		while (it.hasNext())
			System.out.println(it.next());

}
}
```

Problem 3:

Write a program which inputs a text, then implement a code to count the occurrences of each word in a input text and display the word along with corresponding count sorted by the words alphabetically.
(Hint: Use Map as an intermediate collection)
Sample Input
Manoj works at Wipro
Katari works at Wipro
Sureka works at Wipro
Harish works at Wipro
Anitha works at Wipro
Janani works at Wipro
Sample Output
Anitha : 1
Harish : 1

Janani : 1
Katari : 1
Manoj : 1
Sureka : 1
Wipro : 6
at : 6
works : 6

Concept:
Declare Class.
Add method to check occurrence.
Use split method of string to split the string.
Declare loop to iterate in decomposed string.
Compare original string with decomposed substring.
Display the number of occurrence.

Solution:

```
import java.io.*;

class GFG {
static int countOccurences(String str, String word)
{
     String a[] = str.split(" ");
     int count = 0;
   for (int i = 0; i < a.length; i++)
   {
     if (word.equals(a[i]))
      count++;
   }

   return count;
}

public static void main(String args[])
{
   String str = "ABC ABC ABC ABC ABC ABC ABC ABC";
   String word = "AB";
   System.out.println(countOccurences(str, word));
}
}
```