1A - SEQUENTIAL SEARCH

Program

```
public class LinearSearch
 {
          public static void main(String[] args)
         {
                  Scanner sc = new Scanner(System.in);
                  int[] arr = {12, 23, 10, 34, 55, 4, 68, 3, 73, 99};
                   System.out.println("Enter value to search: ");
                   int searchElement = sc.nextInt();
                  int index = linearSearch(arr, searchElement);
                  if(index != -1)
                  {
                  System.out.println("Searched item " + arr[index] + " found at index "+index);
                  }
                  Else
                  {
                   System.out.println("Searched item " + searchElement + " not found in the array");
                   }
           }

           private static int linearSearch(int[] arr, int searchElement)
         {
         for(int i = 0; i < arr.length; i++)
         {
         if(arr[i] == searchElement){
          return i;
     }
    }
    return -1;
  }
}
```

Program

```
public class LinearSearch
{
  public static void main(String[] args)
{
    Scanner sc = new Scanner(System.in);
    int[] arr = {12, 23, 10, 34, 55, 4, 68, 3, 73, 99, 15};
    System.out.println("Enter value to search: ");
    int searchElement = sc.nextInt();
    int index = linearSearch(arr, 0, arr.length - 1, searchElement);
    if(index != -1){
      System.out.println("Searched item " + arr[index] + " found at index "+index);
    }else{
      System.out.println("Searched item " + searchElement + " not found in the array");
    }
  }

  private static int linearSearch(int[] arr, int index, int length, int searchElement){
    // exit condition
    if(index > length)
      return -1;
    // when searched element is found
    if(arr[index] == searchElement){
      return index;
    }
    return linearSearch(arr, index+1, length, searchElement);
  }
}
```

1B - BINARY SEARCH

Program

```
class BinarySearchExample{
 public static void binarySearch(int arr[], int first, int last, int key){
 int mid = (first + last)/2;
while( first <= last ){
 if ( arr[mid] < key ){
first = mid + 1;
}else if ( arr[mid] == key ){
 System.out.println("Element is found at index: " + mid);
break;
 }else{
last = mid - 1;
      }
    mid = (first + last)/2;
  }
  if ( first > last ){
     System.out.println("Element is not found!");
  }
 }
 public static void main(String args[]){
      int arr[] = {10,20,30,40,50};
       int key = 30;
       int last=arr.length-1;
       binarySearch(arr,0,last,key);
 }
}
```

Program

```
class BinarySearchExample1{
   public static int binarySearch(int arr[], int first, int last, int key){
     if (last>=first){
         int mid = first + (last - first)/2;
         if (arr[mid] == key){
         return mid;
         }
         if (arr[mid] > key){
         return binarySearch(arr, first, mid-1, key);//search in left subarray
        }else{
         return binarySearch(arr, mid+1, last, key);//search in right subarray
         }
      }
      return -1;
   }
   public static void main(String args[]){
     int arr[] = {10,20,30,40,50};
     int key = 30;
      int last=arr.length-1;
     int result = binarySearch(arr,0,last,key);
      if (result == -1)
         System.out.println("Element is not found!");
      else
         System.out.println("Element is found at index: "+result);
   }
}
```

1C - SELECTION SORT

Program

```
public class SelectionSortExample {
   public static void selectionSort(int[] arr){
      for (int i = 0; i < arr.length - 1; i++)
      {
         int index = i;
         for (int j = i + 1; j < arr.length; j++){
            if (arr[j] < arr[index]){
               index = j;//searching for lowest index
            }
          }
         int smallerNumber = arr[index];
         arr[index] = arr[i];
         arr[i] = smallerNumber;
      }
   }

   public static void main(String a[]){
      int[] arr1 = {9,14,3,2,43,11,58,22};
      System.out.println("Before Selection Sort");
      for(int i:arr1){
         System.out.print(i+" ");
      }
      System.out.println();

      selectionSort(arr1);//sorting array using selection sort

      System.out.println("After Selection Sort");
      for(int i:arr1){
         System.out.print(i+" ");
      }
   }
}
```

1D - INSERTION SORT

Program

```
Class InsertionSort {
   /*Function to sort array using insertion sort*/
   void sort(int arr[])
   {
      int n = arr.length;
      for (int i = 1; i < n; ++i) {
         int key = arr[i];
         int j = i - 1;

         /* Move elements of arr[0..i-1], that are
           greater than key, to one position ahead
           of their current position */
         while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
         }
         arr[j + 1] = key;
      }
   }

   /* A utility function to print array of size n*/
   static void printArray(int arr[])
   {
      int n = arr.length;
      for (int i = 0; i < n; ++i)
         System.out.print(arr[i] + " ");

      System.out.println();
   }

   // Driver method
   public static void main(String args[])
   {
      int arr[] = { 12, 11, 13, 5, 6 };

      InsertionSort ob = new InsertionSort();
      ob.sort(arr);

      printArray(arr);
   }
}
```

2 - STACK AND QUEUE DATA STRUCTURES USING CLASSES AND OBJECTS

Implementation of Stack Using Array in Java

Program

```
import java.util.*;
public class ArrayStack< E > {
   public static final int CAPACITY = 1000; // default array capacity
   private int topIndex; // index of the top element in stack
   private E[] data; // generic array used for storage
   public ArrayStack() {
      this(CAPACITY);
   } // constructs stack with default capacity

   public ArrayStack(int capacity) { // constructs stack with given capacity
topIndex = -1;
      data = (E[]) new Object[capacity]; // safe cast; compiler may give warning
   }

   public int size() {
      return (topIndex + 1);
   }

   public boolean empty() {
      return (topIndex == -1);
   }
   public void push(E e) throws IllegalStateException {
      if (size() == data.length) throw new IllegalStateException("Stack is full");
      data[++topIndex] = e; // increment topIndex before storing new item
   }

   public E peek() throws EmptyStackException {
      if (empty()) throw new EmptyStackException();
      return data[topIndex];
```

```
    }
    public E pop() throws EmptyStackException {
        if (empty()) throw new EmptyStackException();
        E answer = data[topIndex];
        data[topIndex] = null; // dereference to help garbage collection
topIndex--;
        return answer;
    }

    public static void main(String args[]) {
ArrayStack< Integer >mystack = new ArrayStack<>();
mystack.push(9); //a
mystack.push(3); //b
mystack.push(8); //c
System.out.println("Element at the top is :" + mystack.peek()); //d
System.out.println("Element removed is : " + mystack.pop()); //e
System.out.println("The size of the stack is : " + mystack.size()); //f
System.out.println("Element removed is : " + mystack.pop()); //g
System.out.println("Element at the top is : " + mystack.peek()); //h
mystack.push(10); //i
System.out.println("Stack is empty :  " + mystack.empty()); //j
    /*Note: In output charecters of the comments are written to correspond the
    output, they won't be printed.*/
    }
}
```

Implementation of Stack Using LinkedList in Java

Program

```
import java.util.*;
public class LinkedListStack {
   private Node headNode; // the first node
   private int stackSize;
   // nest class to define linkedlist node
   private class Node {
      int value;
      Node next;
   }
   public LinkedListStack() {
headNode = null;
stackSize = 0;
   }
   // Remove value from the beginning of the list for demonstrating behaviour of stack
   public int pop() throws Exception {
      if (headNode == null) {
         throw new EmptyStackException();
      }
      int value = headNode.value;
headNode = headNode.next;
stackSize--;
      return value;
   }
   // Add value to the beginning of the list to demonstrate behaviour of the stack
   public void push(int value) {
      Node oldHead = headNode;
headNode = new Node();
headNode.value = value;
headNode.next = oldHead;
```

```
stackSize++;
    }
    public int peek() throws Exception {
        if (headNode == null) throw new EmptyStackException();
        else return headNode.value;
    }
    public int size() {
        return stackSize;
    }
    public boolean empty() {
        return stackSize == 0;
    }
    public static void main(String args[]) throws Exception {
LinkedListStackmystack = new LinkedListStack();
mystack.push(9); //a
mystack.push(3); //b
mystack.push(8); //c
System.out.println("Element at the top is :" + mystack.peek()); //d
System.out.println("Element removed is : " + mystack.pop()); //e
System.out.println("The size of the stack is : " + mystack.size()); //f
System.out.println("Element removed is : " + mystack.pop()); //g
System.out.println("Element at the top is : " + mystack.peek()); //h
mystack.push(10);
System.out.println("Stack is empty :  " + mystack.empty()); //i
    }
}
```

Implementation of Queue Using Array in Java

Program

```
import java.util.*;
public class ArrayQueue< E > {
   private E[] data; // generic array used for storage
   // constructors
   private int frontIndex;
   private int queueSize;
   public ArrayQueue(int capacity) { // constructs queue with given capacity
      data = (E[]) new Object[capacity]; // safe cast; compiler may give warning
queueSize = 0; // current number of elements
frontIndex = 0; // index of the front element
   }
   public ArrayQueue() {
      this(1000);
   } // constructs queue with default capacity
   // methods
   /* Returns the number of elements in the queue. */
   public int size() {
      return queueSize;
   }
   /* Tests whether the queue is empty. */
   public booleanisEmpty() {
      return (queueSize == 0);
   }
   /* Inserts an element at the rear of the queue. */
   public void enqueue(E e) throws IllegalStateException {
      if (queueSize == data.length) throw new IllegalStateException("Queue is full");
      int avail = (frontIndex + queueSize) % data.length; // use modular arithmetic
      data[avail] = e;
queueSize++;
```

```
    }
    /* Returns, but does not remove, the first element of the queue (null if empty). */
    public E first() throws IllegalStateException {
        if (queueSize == data.length) throw new IllegalStateException("Queue is empty");
        return data[frontIndex];
    }
    /* Removes and returns the first element of the queue (null if empty). */
    public E dequeue() throws IllegalStateException {
        if (queueSize == data.length) throw new IllegalStateException("Queue is empty");
        E answer = data[frontIndex];
        data[frontIndex] = null; // dereference to help garbage collection
frontIndex = (frontIndex + 1) % data.length;
queueSize--;
        return answer;
    }
    public static void main(String[] args) {
ArrayQueue queue = new ArrayQueue();
queue.enqueue(18); //a
System.out.println("Element at front :  " + queue.first()); //b
System.out.println("Element removed from front : " + queue.dequeue()); //c
System.out.println("Queue is Empty : " + queue.isEmpty()); //d
queue.enqueue(79); //e
queue.enqueue(90); //f
System.out.println("Size of the queue : " + queue.size()); //g
System.out.println("Element removed from front end : " + queue.dequeue());
        //h
    }
}
```

Implementation of Queue Using LinkedList in Java

Program

```
import java.util.*;
public class LinkedListQueue {
   private Node frontNode, rearNode;
   private int queueSize; // queue size
   //linked list node
   private class Node {
      int data;
      Node next;
   }
   //default constructor - initially front & rear are null; size=0; queue is empty
   public LinkedListQueue() {
frontNode = null;
rearNode = null;
queueSize = 0;
   }
   //check if the queue is empty
   public booleanisEmpty() {
      return (queueSize == 0);
   }
   //Remove the item from the front of the queue.
   public int dequeue() throws IllegalStateException {
      if (isEmpty()) throw new IllegalArgumentException("Queue is Empty");
      int data = frontNode.data;
frontNode = frontNode.next;
      if (isEmpty()) {
rearNode = null;
      }
queueSize--;
      return data;
   }
```

```
    public int first() throws IllegalStateException {
        if (isEmpty()) throw new IllegalArgumentException("Queue is Empty");
        return frontNode.data;
    }
    public int size() {
        return queueSize;
    }
    //Add data at the rear of the queue.
    public void enqueue(int data) {
        Node oldRear = rearNode;
rearNode = new Node();
rearNode.data = data;
rearNode.next = null;
        if (isEmpty()) {
frontNode = rearNode;
        } else {
oldRear.next = rearNode;
        }
queueSize++;
    }
    public static void main(String[] args) {
LinkedListQueue queue = new LinkedListQueue();
queue.enqueue(18);
System.out.println("Element at front :  " + queue.first());
System.out.println("Element removed from front : " + queue.dequeue());
System.out.println("Queue is Empty : " + queue.isEmpty());
queue.enqueue(79);
queue.enqueue(90);
System.out.println("Size of the queue : " + queue.size());
System.out.println("Element removed from front end : " + queue.dequeue());
    }
}
```

3 - JAVA APPLICATION USING INHERITANCE

PROGRAM

```
import java.util.*;
class employee{
 int empid;
 long mobile;
String name, address, mailid;
Scanner get = new Scanner(System.in);
void getdata(){
 System.out.println("Enter Name of the Employee");
 name = get.nextLine();
 System.out.println("Enter Mail id");
 mailid = get.nextLine();
 System.out.println("Enter Address of the Employee:");
 address = get.nextLine();
 System.out.println("Enter employee id ");
 empid = get.nextInt();
 System.out.println("Enter Mobile Number");
 mobile = get.nextLong();
 }
 void display()
 {
 System.out.println("Employee Name: "+name);
 System.out.println("Employee id : "+empid);
 System.out.println("Mail id : "+mailid);
 System.out.println("Address: "+address);
 System.out.println("Mobile Number: "+mobile);
 }
}
class programmer extends employee
{
double salary,bp,da,hra,pf,club,net,gross;
```

```
void getprogrammer()
{
System.out.println("Enter basic pay");
bp = get.nextDouble();
}
void calculateprog()
{
da=(0.97*bp);
hra=(0.10*bp);
pf=(0.12*bp);
club=(0.1*bp);
gross=(bp+da+hra);
net=(gross-pf-club);
System.out.println("************************************************");
System.out.println("PAY SLIP FOR PROGRAMMER");
System.out.println("************************************************");
System.out.println("Basic Pay:Rs"+bp);
System.out.println("DA:Rs"+da);
System.out.println("PF:Rs"+pf);
System.out.println("HRA:Rs"+hra);
System.out.println("CLUB:Rs"+club);
System.out.println("GROSS PAY:Rs"+gross);
System.out.println("NET PAY:Rs"+net);
}
}
class asstprofessor extends employee
{
double salary,bp,da,hra,pf,club,net,gross;
void getasst()
{
System.out.println("Enter basic pay");
bp = get.nextDouble();
```

```
}
void calculateasst()
{
da=(0.97*bp);
hra=(0.10*bp);
pf=(0.12*bp);
club=(0.1*bp);
gross=(bp+da+hra);
net=(gross-pf-club);
System.out.println("************************************************");
System.out.println("PAY SLIP FOR ASSISTANT PROFESSOR");
System.out.println("************************************************");
System.out.println("Basic Pay:Rs"+bp);
System.out.println("DA:Rs"+da);
System.out.println("HRA:Rs"+hra);
System.out.println("PF:Rs"+pf);
System.out.println("CLUB:Rs"+club);
System.out.println("GROSS PAY:Rs"+gross);
System.out.println("NET PAY:Rs"+net);
 }
}
class associateprofessor extends employee
{
double salary,bp,da,hra,pf,club,net,gross;
 void getassociate()
{
System.out.println("Enter basic pay");
bp = get.nextDouble();
}
void calculateassociate()
{
da=(0.97*bp);
```

```
hra=(0.10*bp);
pf=(0.12*bp);
club=(0.1*bp);
gross=(bp+da+hra);
net=(gross-pf-club);
System.out.println("************************************************");
System.out.println("PAY SLIP FOR ASSOCIATE PROFESSOR");
System.out.println("************************************************");
System.out.println("Basic Pay:Rs"+bp);
System.out.println("DA:Rs"+da);
System.out.println("HRA:Rs"+hra);
System.out.println("PF:Rs"+pf);
System.out.println("CLUB:Rs"+club);
System.out.println("GROSS PAY:Rs"+gross);
System.out.println("NET PAY:Rs"+net);
}
}
class professor extends employee
{
double salary,bp,da,hra,pf,club,net,gross;
void getprofessor()
{
System.out.println("Enter basic pay");
bp = get.nextDouble();
}
void calculateprofessor()
{
da=(0.97*bp);
hra=(0.10*bp);
pf=(0.12*bp);
club=(0.1*bp);
gross=(bp+da+hra);
```

```
net=(gross-pf-club);
System.out.println("************************************************");
System.out.println("PAY SLIP FOR PROFESSOR");
System.out.println("************************************************");
System.out.println("Basic Pay:Rs"+bp);
System.out.println("DA:Rs"+da);
System.out.println("HRA:Rs"+hra);
System.out.println("PF:Rs"+pf);
System.out.println("CLUB:Rs"+club);
System.out.println("GROSS PAY:Rs"+gross);
System.out.println("NET PAY:Rs"+net);
}
}
class salary
{
public static void main(String args[])
{
int choice,cont;
do
{
System.out.println("PAYROLL");
System.out.println(" 1.PROGRAMMER \t 2.ASSISTANT PROFESSOR \t 3.ASSOCIATE
PROFESSOR \t 4.PROFESSOR ");
Scanner c = new Scanner(System.in);
choice=c.nextInt();
switch(choice){
case 1:{
programmer p=new programmer();
p.getdata();
p.getprogrammer();
p.display();
p.calculateprog();
```

```
break;
}
case 2:{
asstprofessor asst=new asstprofessor();
asst.getdata();
asst.getasst();
asst.display();
asst.calculateasst();
break;
}
case 3:{
associateprofessor asso=new associateprofessor();
asso.getdata();
asso.getassociate();
asso.display();
asso.calculateassociate();
break;
}
case 4:{
professor prof=new professor();
prof.getdata();
prof.getprofessor();
prof.display();
prof.calculateprofessor();
break;
}
}
System.out.println("Do u want to continue 0 to quit and  1 to continue ");
cont=c.nextInt();
}while(cont==1);
}
}
```

4 - JAVA PROGRAM USING ABSTRACT CLASS

PROGRAM

```
import java.util.*;
abstract class shape{
 int a,b;
 abstract public void printarea();
}
class rectangle extends shape{
public int area_rect;
public void printarea()
{
  Scanner s=new Scanner(System.in);
  System.out.println("enter the length and breadth of rectangle");
  a=s.nextInt();
  b=s.nextInt();
  area_rect=a*b;
  System.out.println("Length of rectangle "+a +"breadth of rectangle "+b);
  System.out.println("The area ofrectangle is:"+area_rect);
 }
 }
class triangle extends shape{
 double area_tri;
 public void printarea()
{
Scanner s=new Scanner(System.in);
System.out.println("enter the base and height of triangle");
a=s.nextInt();
b=s.nextInt();
System.out.println("Base of triangle "+a +"height of triangle  "+b);
area_tri=(0.5*a*b);
```

```
System.out.println("The area of triangle is:"+area_tri);
}
}
class circle extends shape{
double area_circle;
public void printarea()
{
Scanner s=new Scanner(System.in);
System.out.println("enter the radius of circle");
a=s.nextInt();
 area_circle=(3.14*a*a);
 System.out.println("Radius of circle"+a);
 System.out.println("The area of circle is:"+area_circle);
 }
}
public class shapeclass
{
 public static void main(String[] args)
{
rectangle r=new rectangle();
r.printarea();
 triangle t=new triangle();
 t.printarea();
circle r1=new circle();
r1.printarea();
}
}
```

5 - PROGRAM FOR STACK ADT USING INTERFACE

PROGRAM

```
import java.io.*;
interface stackoperation{
 public void push(int i);
 public void pop();
}
class Astack implements stackoperation{
int stack[]=new int[5];
int top=-1;
int i;
public void push(int item)
{
if(top>=4)
{
System.out.println("overflow");
}
else
{
top=top+1;
stack[top]=item;
System.out.println("item pushed"+stack[top]);
}
}
public void pop()
{
if(top<0)
System.out.println("underflow");
else
{
```

```
System.out.println("item popped"+stack[top]);
top=top-1;
}
}
public void display()
{
if(top<0)
System.out.println("No Element in stack");
else
{
for(i=0;i<=top;i++)
System.out.println("element:"+stack[i]);
}
}
}
class teststack{
public static void main(String args[])throws IOException
{
int ch,c;
int i;
Astack s=new Astack();
DataInputStream in=new DataInputStream(System.in);
do{
try{
System.out.println("ARRAY STACK");
System.out.println("1.push  2.pop  3.display  4.exit");
System.out.println("enter ur choice:");
ch=Integer.parseInt(in.readLine());
switch(ch)
{
case 1:
System.out.println("enter the value to push:");
```

```
i=Integer.parseInt(in.readLine());
s.push(i);
break;
case 2:
s.pop();
break;
case 3:
System.out.println("the elements are:");
s.display();
break;
case 4:
break;
}
}
catch(IOException e)
{
System.out.println("io error");
}
System.out.println("Do u want to continue 0 to quit and  1 to continue ");
c=Integer.parseInt(in.readLine());
}while(c==1);
}
}
```

6 - PROGRAM TO IMPLEMENT EXCEPTION HANDLING AND CREATION OF USER DEFINED EXCEPTIONS.

(a)PROGRAM

```
import java.util.Scanner;
 class NegativeAmtException extends Exception{
 String msg;
 NegativeAmtException(String msg){
      this.msg=msg;
 }
 public String toString() {
       return msg;
 }
}
public class userdefined {
 public static void main(String[] args){
Scanner s=new Scanner(System.in);
System.out.print("Enter Amount:");
int a=s.nextInt();
try{
if(a<0)
 {
throw new NegativeAmtException("Invalid Amount");
 }
 System.out.println("Amount Deposited");
 }
catch(NegativeAmtException e){
 System.out.println(e);
}
}
}
```

(b) PROGRAM

```
class MyException extends Exception{
String str1;
MyException(String str2)
 {
str1=str2;
  }
  public String toString()
 {
return ("MyException Occurred: "+str1) ;
 }
}
class example
{
 public static void main(String args[])
{
try
{
System.out.println("Starting of try block");
throw new MyException("This is My error Message");
}
catch(MyException exp)
{
System.out.println("Catch Block") ;
System.out.println(exp) ;
}
}
}
```

7 - JAVA PROGRAM THAT IMPLEMENTS MULTI-THREADED APPLICATION

PROGRAM

```
import java.util.*;
class even implements Runnable{
public int x;
public even(int x){
this.x = x;
}
public void run(){
System.out.println("New Thread "+ x +" is EVEN and Square of " + x + " is: " + x * x);
}
}
class odd implements Runnable{
public int x;
public odd(int x){
this.x = x;
}
public void run(){
System.out.println("New Thread "+ x +" is ODD and Cube of " + x + " is: " + x * x * x);
}
}
class A extends Thread{
public void run(){
int num = 0;
Random r = new Random();
try
{
for (int i = 0; i < 5; i++)
{
num = r.nextInt(100);
System.out.println("Main Thread and Generated Number is " + num);
```

```
if (num % 2 == 0)
{
Thread t1 = new Thread(new even(num));
t1.start();
}
else
{
Thread t2 = new Thread(new odd(num));
t2.start();
}
Thread.sleep(1000);
System.out.println("-------------------------------------");
}
}
catch (Exception ex){
System.out.println(ex.getMessage());
}
}
}
public class multithreadprog{
public static void main(String[] args){
A a = new A();
a.start();
}
}
```

8 - WRITE A PROGRAM TO PERFORM FILE OPERATIONS

PROGRAM

```
import java.io.*;
import java.util.*;
class filedemo
{
public static void main(String args[])
{
String filename;
Scanner s=new Scanner(System.in);
System.out.println("Enter the file name ");
filename=s.nextLine();
File f1=new File(filename);
System.out.println("******************");
System.out.println("FILE INFORMATION");
System.out.println("******************");
System.out.println("NAME OF THE FILE "+f1.getName());
System.out.println("PATH OF THE FILE "+f1.getPath());
System.out.println("PARENT"+f1.getParent());
if(f1.exists())
System.out.println("THE FILE EXISTS ");
else
System.out.println("THE FILE DOES NOT ExISTS ");
if(f1.canRead())
System.out.println("THE FILE CAN BE READ ");
else
System.out.println("THE FILE CANNOT BE READ  ");
if(f1.canWrite())
System.out.println("WRITE OPERATION IS PERMITTED");
else
```

```
System.out.println("WRITE OPERATION IS NOT PERMITTED");
if(f1.isDirectory())
System.out.println("IT IS A DIRECTORY ");
else
System.out.println("NOT A DIRECTORY");
if(f1.isFile())
System.out.println("IT IS A FILE ");
else
System.out.println("NOT A FILE");
System.out.println("File last modified "+ f1.lastModified());
System.out.println("LENGTH OF THE FILE  "+f1.length());
System.out.println("FILE DELETED "+f1.delete());
}
}
```

9 - DEVELOP APPLICATIONS TO DEMONSTRATE THE FEATURES OF GENERIC CLASSES

PROGRAM

```
class MyClass<T extends Comparable<T>>
{
T[] vals;
MyClass(T[] o)
{
vals = o;
}
public T min()
{
T v = vals[0];
for(int i=1; i < vals.length; i++)
if(vals[i].compareTo(v) < 0)
v = vals[i];
return v;
}
public T max()
{
T v = vals[0];
for(int i=1; i < vals.length;i++)
if(vals[i].compareTo(v) > 0)
v = vals[i];
return v;
}
}
class gendemo
{
public static void main(String args[])
{
int i;
```

```
Integer inums[]={10,2,5,4,6,1};
Character chs[]={'v','p','s','a','n','h'};
Double d[]={20.2,45.4,71.6,88.3,54.6,10.4};
MyClass<Integer> iob = new MyClass<Integer>(inums);
MyClass<Character> cob = new MyClass<Character>(chs);
MyClass<Double>dob = new MyClass<Double>(d);
System.out.println("Max value in inums: " + iob.max());
System.out.println("Min value in inums: " + iob.min());
System.out.println("Max value in chs: " + cob.max());
System.out.println("Min value in chs: " + cob.min());
System.out.println("Max value in chs: " + dob.max());
System.out.println("Min value in chs: " + dob.min());
}
}
```

10 - DEVELOP APPLICATIONS USING JAVAFX CONTROLS, LAYOUTS AND MENUS.

PROGRAM

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.event.*;
public class ScientificCalculator extends JFrame implements ActionListener
{
JTextField tfield;
double temp, temp1, result, a;
static double m1, m2;
int k = 1, x = 0, y = 0, z = 0;
char ch;
JButton b1, b2, b3, b4, b5, b6, b7, b8, b9, zero, clr, pow2, pow3, exp,
fac, plus, min, div, log, rec, mul, eq, addSub, dot, mr, mc, mp,
mm, sqrt, sin, cos, tan;
Container cont;
JPanel textPanel, buttonpanel;
ScientificCalculator()
{
cont = getContentPane();
cont.setLayout(new BorderLayout());
JPanel textpanel = new JPanel();
tfield = new JTextField(25);
tfield.setHorizontalAlignment(SwingConstants.RIGHT);
tfield.addKeyListener(new KeyAdapter() {
public void keyTyped(KeyEvent keyevent) {
char c = keyevent.getKeyChar();
if (c >= '0' && c <= '9') {
```

```
}
else
{
keyevent.consume();
}
}
});
textpanel.add(tfield);
buttonpanel = new JPanel();
buttonpanel.setLayout(new GridLayout(8, 4, 2, 2));
boolean t = true;
mr = new JButton("MR");
buttonpanel.add(mr);
mr.addActionListener(this);
mc = new JButton("MC");
buttonpanel.add(mc);
mc.addActionListener(this);
mp = new JButton("M+");
buttonpanel.add(mp);
mp.addActionListener(this);
mm = new JButton("M-");
buttonpanel.add(mm);
mm.addActionListener(this);
b1 = new JButton("1");
buttonpanel.add(b1);
b1.addActionListener(this);
b2 = new JButton("2");
buttonpanel.add(b2);
b2.addActionListener(this);
b3 = new JButton("3");
buttonpanel.add(b3);
b3.addActionListener(this);
```

```
b4 = new JButton("4");
buttonpanel.add(b4);
b4.addActionListener(this);
b5 = new JButton("5");
buttonpanel.add(b5);
b5.addActionListener(this);
b6 = new JButton("6");
buttonpanel.add(b6);
b6.addActionListener(this);
b7 = new JButton("7");
buttonpanel.add(b7);
b7.addActionListener(this);
b8 = new JButton("8");
buttonpanel.add(b8);
b8.addActionListener(this);
b9 = new JButton("9");
buttonpanel.add(b9);
b9.addActionListener(this);
zero = new JButton("0");
buttonpanel.add(zero);
zero.addActionListener(this);
plus = new JButton("+");
buttonpanel.add(plus);
plus.addActionListener(this);
min = new JButton("-");
buttonpanel.add(min);
min.addActionListener(this);
mul = new JButton("*");
buttonpanel.add(mul);
mul.addActionListener(this);
div = new JButton("/");
div.addActionListener(this);
```

```
buttonpanel.add(div);
addSub = new JButton("+/-");
buttonpanel.add(addSub);
addSub.addActionListener(this);
dot = new JButton(".");
buttonpanel.add(dot);
dot.addActionListener(this);
eq = new JButton("=");
buttonpanel.add(eq);
eq.addActionListener(this);
rec = new JButton("1/x");
buttonpanel.add(rec);
rec.addActionListener(this);
sqrt = new JButton("Sqrt");
buttonpanel.add(sqrt);
sqrt.addActionListener(this);
log = new JButton("log");
buttonpanel.add(log);
log.addActionListener(this);
sin = new JButton("SIN");
buttonpanel.add(sin);
sin.addActionListener(this);
cos = new JButton("COS");
buttonpanel.add(cos);
cos.addActionListener(this);
tan = new JButton("TAN");
buttonpanel.add(tan);
tan.addActionListener(this);
pow2 = new JButton("x^2");
buttonpanel.add(pow2);
pow2.addActionListener(this);
pow3 = new JButton("x^3");
```

```
buttonpanel.add(pow3);
pow3.addActionListener(this);
exp = new JButton("Exp");
exp.addActionListener(this);
buttonpanel.add(exp);
fac = new JButton("n!");
fac.addActionListener(this);
buttonpanel.add(fac);
clr = new JButton("AC");
buttonpanel.add(clr);
clr.addActionListener(this);
cont.add("Center", buttonpanel);
cont.add("North", textpanel);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
public void actionPerformed(ActionEvent e)
{
String s = e.getActionCommand();
if (s.equals("1"))
{
if (z == 0)
{
tfield.setText(tfield.getText() + "1");
}
else
{
tfield.setText("");
tfield.setText(tfield.getText() + "1");
z = 0;
}
}
if (s.equals("2")) {
```

```
if (z == 0) {
tfield.setText(tfield.getText() + "2");
}
else
{
tfield.setText("");
tfield.setText(tfield.getText() + "2");
z = 0;
}
}
if (s.equals("3")) {
if (z == 0) {
tfield.setText(tfield.getText() + "3");
}
else
{
tfield.setText("");
tfield.setText(tfield.getText() + "3");
z = 0;
}
}
if (s.equals("4")) {
if (z == 0) {
tfield.setText(tfield.getText() + "4");
}
else
{
tfield.setText("");
tfield.setText(tfield.getText() + "4");
z = 0;
}
}
```

```
if (s.equals("5")) {
if (z == 0) {
tfield.setText(tfield.getText() + "5");
}
else
{
tfield.setText("");
tfield.setText(tfield.getText() + "5");
z = 0;
}
}
if (s.equals("6")) {
if (z == 0) {
tfield.setText(tfield.getText() + "6");
}
else
 {
tfield.setText("");
tfield.setText(tfield.getText() + "6");
z = 0;
}
}
if (s.equals("7")) {
if (z == 0) {
tfield.setText(tfield.getText() + "7");
}
else
{
tfield.setText("");
tfield.setText(tfield.getText() + "7");
z = 0;
}
```

```
}
if (s.equals("8")) {
if (z == 0) {
tfield.setText(tfield.getText() + "8");
}
else
{
tfield.setText("");
tfield.setText(tfield.getText() + "8");
z = 0;
}
}
if (s.equals("9")) {
if (z == 0) {
tfield.setText(tfield.getText() + "9");
}
else
{
tfield.setText("");
tfield.setText(tfield.getText() + "9");
z = 0;
}
}
if (s.equals("0"))
{
if (z == 0) {
tfield.setText(tfield.getText() + "0");
}
else
{
tfield.setText("");
tfield.setText(tfield.getText() + "0");
```

```
z = 0;
}
}
if (s.equals("AC")) {
tfield.setText("");
x = 0;
y = 0;
z = 0;
}
if (s.equals("log"))
{
if (tfield.getText().equals("")) {
tfield.setText("");
}
else
 {
a = Math.log(Double.parseDouble(tfield.getText()));
tfield.setText("");
tfield.setText(tfield.getText() + a);
}
}
if (s.equals("1/x")) {
if (tfield.getText().equals("")) {
tfield.setText("");
}
else
{
a = 1 / Double.parseDouble(tfield.getText());
tfield.setText("");
tfield.setText(tfield.getText() + a);
}
}
```

```
if (s.equals("Exp")) {
if (tfield.getText().equals("")) {
tfield.setText("");
}
else
{
a = Math.exp(Double.parseDouble(tfield.getText()));
tfield.setText("");
tfield.setText(tfield.getText() + a);
}
}
if (s.equals("x^2")) {
if (tfield.getText().equals("")) {
tfield.setText("");
}
else
{
a = Math.pow(Double.parseDouble(tfield.getText()), 2);
tfield.setText("");
tfield.setText(tfield.getText() + a);
}
}
if (s.equals("x^3")) {
if (tfield.getText().equals("")) {
tfield.setText("");
}
else
{
a = Math.pow(Double.parseDouble(tfield.getText()), 3);
tfield.setText("");
tfield.setText(tfield.getText() + a);
}
```

```
}
if (s.equals("+/-")) {
if (x == 0) {
tfield.setText("-" + tfield.getText());
x = 1;
}
else
{
tfield.setText(tfield.getText());
}
}
if (s.equals(".")) {
if (y == 0) {
tfield.setText(tfield.getText() + ".");
y = 1;
}
else
{
tfield.setText(tfield.getText());
}
}
if (s.equals("+"))
{
if (tfield.getText().equals(""))
{
tfield.setText("");
temp = 0;
ch = '+';
}
else
{
temp = Double.parseDouble(tfield.getText());
```

```
tfield.setText("");
ch = '+';
y = 0;
x = 0;
}
tfield.requestFocus();
}
if (s.equals("-"))
{
if (tfield.getText().equals(""))
{
tfield.setText("");
temp = 0;
ch = '-';
}
 else
 {
x = 0;
y = 0;
temp = Double.parseDouble(tfield.getText());
tfield.setText("");
ch = '-';
}
tfield.requestFocus();
}
if (s.equals("/")) {
if (tfield.getText().equals(""))
{
tfield.setText("");
temp = 1;
ch = '/';
}
```

```
else
{
x = 0;
y = 0;
temp = Double.parseDouble(tfield.getText());
ch = '/';
tfield.setText("");
}
tfield.requestFocus();
}
if (s.equals("*")) {
if (tfield.getText().equals(""))
{
tfield.setText("");
temp = 1;
ch = '*';
}
else
{
x = 0;
y = 0;
temp = Double.parseDouble(tfield.getText());
ch = '*';
tfield.setText("");
}
tfield.requestFocus();
}
if (s.equals("MC"))
{
m1 = 0;
tfield.setText("");
}
```

```
if (s.equals("MR"))
{
tfield.setText("");
tfield.setText(tfield.getText() + m1);
}
if (s.equals("M+"))
{
if (k == 1) {
m1 = Double.parseDouble(tfield.getText());
k++;
}
else
{
m1 += Double.parseDouble(tfield.getText());
tfield.setText("" + m1);
}
}
if (s.equals("M-"))
{
if (k == 1) {
m1 = Double.parseDouble(tfield.getText());
k++;
}
else
 {
m1 -= Double.parseDouble(tfield.getText());
tfield.setText("" + m1);
}
}
if (s.equals("Sqrt"))
{
if (tfield.getText().equals(""))
```

```
{
tfield.setText("");
}
else
{
a = Math.sqrt(Double.parseDouble(tfield.getText()));
tfield.setText("");
field.setText(tfield.getText() + a);
}
}
if (s.equals("SIN"))
{
if (tfield.getText().equals(""))
{
tfield.setText("");
}
else
{
a = Math.sin(Double.parseDouble(tfield.getText()));
tfield.setText("");
tfield.setText(tfield.getText() + a);
}
}
if (s.equals("COS"))
{
if (tfield.getText().equals(""))
{
tfield.setText("");
}
else
{
a = Math.cos(Double.parseDouble(tfield.getText()));
```

```
tfield.setText("");
tfield.setText(tfield.getText() + a);
}
}
if (s.equals("TAN")) {
if (tfield.getText().equals("")) {
tfield.setText("");
}
else
{
a = Math.tan(Double.parseDouble(tfield.getText()));
tfield.setText("");
tfield.setText(tfield.getText() + a);
}
}
if (s.equals("="))
{
if (tfield.getText().equals(""))
{
tfield.setText("");
}
else
{
temp1 = Double.parseDouble(tfield.getText());
switch (ch)
{
case '+':
result = temp + temp1;
break;
case '-':
result = temp - temp1;
break;
```

```
case '/':
result = temp / temp1;
break;
case '*':
result = temp * temp1;
break;
}
tfield.setText("");
tfield.setText(tfield.getText() + result);
z = 1;
}
}
if (s.equals("n!"))
{
if (tfield.getText().equals(""))
{
tfield.setText("");
}
else
{
a = fact(Double.parseDouble(tfield.getText()));
tfield.setText("");
tfield.setText(tfield.getText() + a);
}
}
tfield.requestFocus();
}
double fact(double x)
{
int er = 0;
if (x < 0)
{
```

```
er = 20;
return 0;
}
double i, s = 1;
for (i = 2; i <= x; i += 1.0)
s *= i;
return s;
}
public static void main(String args[]) {
try{
UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
}
catch (Exception e) {
}
ScientificCalculator f = new ScientificCalculator();
f.setTitle("ScientificCalculator");
f.pack();
f.setVisible(true);
}
}
```

11 - MINI PROJECT FOR LIBRARY MANAGEMENT SYSTEM

CODING

Logon.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;
public class Logon extends JFrame implements ActionListener
{
int fl=1;
JPanel pLog = new JPanel();
JLabel lbUser, lbPass;
JTextField txtUser;
JPasswordField txtPass;
JButton btnOk, btnCancel;
Connection con;
public String user;
public Logon ()
{
super ("Library Management System.");
setSize (275, 300);
addWindowListener (new WindowAdapter ()
{
public void windowClosing (WindowEvent we) {
setVisible (false);
dispose();
System.exit (0);
}
}
);
pLog.setLayout (null);
lbUser = new JLabel ("Username:");
lbUser.setForeground (Color.black);
lbUser.setBounds (20, 15, 75, 25);
```

```
lbPass = new JLabel ("Password:");
lbPass.setForeground (Color.BLACK);
lbPass.setBounds (20, 50, 75, 25);
txtUser = new JTextField ();
txtUser.setBounds (100, 15, 150, 25);
txtPass = new JPasswordField ();
txtPass.setBounds (100, 50, 150, 25);
//Setting the Form's Buttons.
btnOk = new JButton ("OK");
btnOk.setBounds (20, 90, 100, 25);
btnOk.addActionListener (this);
btnCancel = new JButton ("Cancel");
btnCancel.setBounds (150, 90, 100, 25);
btnCancel.addActionListener (this);
pLog.add (lbUser);
pLog.add (lbPass);
pLog.add (txtUser);
pLog.add (txtPass);
pLog.add (btnOk);
pLog.add (btnCancel);
getContentPane().add (pLog);
//Opening the Database.
try
{
Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
String loc = "jdbc:odbc:temp1";
con = DriverManager.getConnection (loc);
}
catch (ClassNotFoundException cnf)  {
JOptionPane.showMessageDialog (null, "Driver not Loaded...");
System.exit (0);
}
catch (SQLException sqlex) {
JOptionPane.showMessageDialog (null, "Unable to Connect to Database...");
System.exit (0);
```

```
}
//Showing The Logon Form.
setVisible (true);
}
public void actionPerformed (ActionEvent ae)
{
Object obj = ae.getSource();
if (obj == btnOk)
{
String password = new String (txtPass.getPassword());
if (txtUser.getText().equals (""))
{
JOptionPane.showMessageDialog (this, "Provide Username to Logon.");
txtUser.requestFocus();
}
else if (password.equals (""))
{
txtPass.requestFocus();
JOptionPane.showMessageDialog (null,"Provide Password to Logon.");
}
else
{
String pass;
boolean verify = false;
if(fl==1)
{
if(txtUser.getText().equals("Admin")&&password.equals("admin"))
{
verify=true;
new LibrarySystem(1,1,con);
setVisible(false);
dispose();
}
}
else
```

```
{
String tablename=null;
if(fl==2) tablename="Clerks";
else if(fl==3)tablename="Members";
try {        //SELECT Query to Retrieved the Record.
String query = "SELECT * FROM " + tablename + " WHERE id = " + Integer.parseInt(txtUser.getText());
Statement st = con.createStatement ();
ResultSet rs = st.executeQuery (query);
rs.next();
user = rs.getString ("id");
pass = rs.getString ("Password");
if (txtUser.getText().equals (user) && password.equals (pass))
{
verify = true;
new LibrarySystem (fl,Integer.parseInt(txtUser.getText()), con);
setVisible (false);
dispose();
}
else
{
verify = false;
txtUser.setText ("");
txtPass.setText ("");
txtUser.requestFocus ();
}
}
catch (Exception sqlex)
 {
if (verify == false)
{
txtUser.setText ("");
txtPass.setText ("");
txtUser.requestFocus ();
}
}
```

```
}
}
}
else if (obj == btnCancel)
{
setVisible (false);
dispose();
System.exit (0);
}
}
public static void main(String args[])
{
Logon start=new Logon();
}
}
class FrmSplash extends JWindow implements Runnable
{
Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
public void run()
{
setSize(275,300);
setVisible(true);
}
}
```

AddBCat.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;

public class AddBCat extends JInternalFrame implements ActionListener
{
JPanel pNew = new JPanel();
```

```
JLabel lbUser;
JTextField txtUser;
JButton btnOk, btnCancel;

private Statement st;
public AddBCat (Connection con)
{
super ("New Book Category", false, true, false, true);
setSize (280, 175);
lbUser = new JLabel ("Category:");
lbUser.setForeground (Color.black);
lbUser.setBounds (20, 20, 100, 25);
txtUser = new JTextField ();
txtUser.setBounds (100, 20, 150, 25);
btnOk = new JButton ("OK");
btnOk.setBounds (20, 100, 100, 25);
btnOk.addActionListener (this);
btnCancel = new JButton ("Cancel");
btnCancel.setBounds (150, 100, 100, 25);
btnCancel.addActionListener (this);
pNew.setLayout (null);
pNew.add (lbUser);
pNew.add (txtUser);
pNew.add (btnOk);
pNew.add (btnCancel);
getContentPane().add (pNew);
try
{
st = con.createStatement ();
}
catch (SQLException sqlex)
{
JOptionPane.showMessageDialog (null, "A Problem Occurs While Loading the Form.");
dispose ();
}
```

```
setVisible (true);
}
public void actionPerformed (ActionEvent ae)
{
Object obj = ae.getSource();
if (obj == btnOk)
{
if (txtUser.getText().equals ("")) {
txtUser.requestFocus();
JOptionPane.showMessageDialog (this, "Category not Provided.");
}
else
{
try
{
String id= txtUser.getText();
String q = "SELECT * FROM BCat ";
ResultSet rs = st.executeQuery (q);
int fl=0;
while(rs.next())
{
String memberNo = rs.getString ("Cat");
if(id.equals(memberNo))
{
JOptionPane.showMessageDialog(this,"Already existing Category");
txtUser.setText("");
txtUser.requestFocus();
fl=1;
break;
}
}
rs.close();
if(fl==0){
q = "INSERT INTO BCat " + "VALUES ('" + txtUser.getText() + "')";
int result = st.executeUpdate (q);
```

```
if (result == 1) {
JOptionPane.showMessageDialog (this, "New Category Created.");
txtUser.setText ("");
txtUser.requestFocus ();
}
else
{
JOptionPane.showMessageDialog (this, "Problem while Creating. ");
txtUser.setText ("");
txtUser.requestFocus ();
}
}
}
catch (SQLException sqlex)
{
JOptionPane.showMessageDialog (this, "Problem while Creating excep.");
}
}
}
if (obj == btnCancel) {
setVisible (false);
dispose();
}
}
}
```

AddBook.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;

public class AddBook extends JInternalFrame implements ActionListener, FocusListener
{
```

```
JPanel pBook = new JPanel ();
JLabel lbBookId, lbBookName, lbBookAuthor, lbBookRef, lbBookCategory;
JTextField txtBookId, txtBookName, txtBookAuthor;
JComboBox cboBookCategory;
JButton btnOk, btnCancel;
JRadioButton rby,rbn;
ButtonGroup bg;
String[] cn =new String[100];
Statement st;
long id = 0;
int i,j,ref=0;
public AddBook (Connection con)
{
super ("Add New Book", false ,true, true, true);
setSize (325, 250);
lbBookId = new JLabel ("Book Id:");
lbBookId.setForeground (Color.black);
lbBookId.setBounds (15, 15, 100, 20);
lbBookName = new JLabel ("Book Name:");
lbBookName.setForeground (Color.black);
lbBookName.setBounds (15, 45, 100, 20);
lbBookAuthor = new JLabel ("Book Author:");
lbBookAuthor.setForeground (Color.black);
lbBookAuthor.setBounds (15, 75, 100, 20);
lbBookRef = new JLabel ("Reference:");
lbBookRef.setForeground (Color.black);
lbBookRef.setBounds (15, 105, 100, 20);
lbBookCategory = new JLabel ("Book Category:");
lbBookCategory.setForeground (Color.black);
lbBookCategory.setBounds (15, 135, 100, 20);
txtBookId = new JTextField ();
txtBookId.setHorizontalAlignment (JTextField.RIGHT);
txtBookId.addFocusListener (this);
txtBookId.setBounds (120, 15, 175, 25);
txtBookName = new JTextField ();
```

```
txtBookName.setBounds (120, 45, 175, 25);
txtBookAuthor = new JTextField ();
txtBookAuthor.setBounds (120, 75, 175, 25);
rby=new JRadioButton("yes");
rby.addActionListener(this);
rby.setBounds(120,105,60,25);
rbn=new JRadioButton("no");
rbn.addActionListener(this);
rbn.setBounds(180,105,60,25);
bg = new ButtonGroup();
bg.add(rby);
bg.add(rbn);
rbn.setSelected(true);
cboBookCategory = new JComboBox();
cboBookCategory.setBounds (120, 135, 175, 25);
btnOk = new JButton ("OK");
btnOk.setBounds (50, 175, 100, 25);
btnOk.addActionListener (this);
btnCancel = new JButton ("Cancel");
btnCancel.setBounds (170, 175, 100, 25);
btnCancel.addActionListener (this);
txtBookId.addKeyListener (new KeyAdapter ()
{
public void keyTyped (KeyEvent ke)
{
char c = ke.getKeyChar ();
if (! ((Character.isDigit (c)) || (c == KeyEvent.VK_BACK_SPACE)))
{
getToolkit().beep ();
ke.consume ();
}
}
}
);
txtBookAuthor.addKeyListener (new KeyAdapter ()
```

```
{
public void keyTyped (KeyEvent ke)
{
char c = ke.getKeyChar ();
if (! ((Character.isLetter (c)) || (c == KeyEvent.VK_BACK_SPACE)||(c == KeyEvent.VK_SPACE)))
{
getToolkit().beep ();
ke.consume ();
}
}
}
);
pBook.setLayout (null);
pBook.add (lbBookId);
pBook.add (lbBookName);
pBook.add (lbBookAuthor);
pBook.add (lbBookRef);
pBook.add (lbBookCategory);
pBook.add (txtBookId);
pBook.add (txtBookName);
pBook.add (txtBookAuthor);
pBook.add (rby);
pBook.add (rbn);
pBook.add (cboBookCategory);
pBook.add (btnOk);
pBook.add (btnCancel);
getContentPane().add (pBook, BorderLayout.CENTER);
try {
i=0;
st = con.createStatement ();
ResultSet rs=st.executeQuery("Select * from BCat");
while(rs.next())
{
cn[i]=rs.getString(1);
i++;
```

```
}
for(j=0;j<i;j++)
{
cboBookCategory.addItem(cn[j]);
}
cboBookCategory.addActionListener(this);
cboBookCategory.setSelectedItem(cn[0]);
rs.close();
}
catch (SQLException sqlex)
{
JOptionPane.showMessageDialog (null, "A Problem Occurs While Loading Form.");
dispose ();
}
setVisible (true);
}
public void actionPerformed (ActionEvent ae)
{
Object obj = ae.getSource();
if (obj == btnOk)
{
if (txtBookId.getText().equals (""))
{
JOptionPane.showMessageDialog (this, "Book's Id not Provided.");
txtBookId.requestFocus ();
}
else if (txtBookName.getText().equals ("")) {
JOptionPane.showMessageDialog (this, "Book's Name not Provided.");
txtBookName.requestFocus ();
}
else if (txtBookAuthor.getText().equals (""))
{
JOptionPane.showMessageDialog (this, "Book's Author Name not Provided.");
txtBookAuthor.requestFocus ();
}
```

```
else

{

try {

int x = 0;

String s8 = x+"/"+x+"/"+x ;


//INSERT Query to Add Book Record in Table.

/* String q = "INSERT INTO Books " +"VALUES (" + id + ", '" + txtBookName.getText() + "', '" + txtBookAuthor.getText() + "', " +
ref + ", '" + cboBookCategory.getSelectedItem() + "' ,"+ 0 + ", '" + s8 + "', '" + s8 + ")"; */


int result = st.executeUpdate ("Insert into Books values("+ id +",'" + txtBookName.getText() +"','" + txtBookAuthor.getText() +"', "
+ ref + ", '" + cboBookCategory.getSelectedItem().toString() +"', " + 0 + ", '"+ s8 +"' ,'"+ s8 + "')");

//Running Query.

if (result == 1) {

//If Query Successful.

JOptionPane.showMessageDialog (this, "Record has been Saved.");

txtClear ();                                    //Clearing the TextFields.

}

else

 {

//If Query Failed.

OptionPane.showMessageDialog (this, "Problem while Saving the Record.");

}

}

catch (SQLException sqlex) {

JOptionPane.showMessageDialog (this, "Problem while Saving the Record Excep.");

}

}

}

if (obj == btnCancel) {        /If Cancel Button Pressed Unload the From.

setVisible (false);

dispose();


}

if(obj==rby)

{
```

```
ref=1;
}
else if(obj==rbn)
{
ref=0;
}
}
public void focusGained (FocusEvent fe) { }
public void focusLost (FocusEvent fe) {
if (txtBookId.getText().equals ("")) {
}
else {
id = Integer.parseInt (txtBookId.getText ());
long bookNo;
boolean found = false;
try {
String q = "SELECT * FROM Books WHERE BId = " + id + "";
ResultSet rs = st.executeQuery (q);
rs.next ();
bookNo = rs.getLong ("BId");
if (bookNo == id) {
found = true;
txtClear ();
JOptionPane.showMessageDialog (this, id + " is already assigned.");
}
else {
found = false;
}
}
catch (SQLException sqlex)
{
}
}
}
private void txtClear () {
```

```
txtBookId.setText ("");
txtBookName.setText ("");
txtBookAuthor.setText ("");
cboBookCategory.setSelectedIndex(0);
txtBookId.requestFocus ();
}
}
```

AddMCat.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;

public class AddMCat extends JInternalFrame implements ActionListener {
JPanel pNew = new JPanel();
JLabel lbUser,lbDate,lbBooks;
JTextField txtUser,txtDate,txtBooks;
JButton btnOk, btnCancel;
private Statement st;
public AddMCat (Connection con) {
super ("New Member Category", false, true, false, true);
setSize (280, 200);
lbUser = new JLabel ("Category:");
lbUser.setForeground (Color.black);
lbUser.setBounds (20, 20, 100, 25);
lbDate = new JLabel ("Days Issued:");
lbDate.setForeground (Color.black);
lbDate.setBounds (20, 55, 100, 25);
lbBooks= new JLabel ("No. of Books");
lbBooks.setForeground (Color.black);
lbBooks.setBounds (20, 90, 100, 25);
txtUser = new JTextField ();
txtUser.setBounds (100, 20, 150, 25);
```

```
txtDate = new JTextField ();
txtDate.setBounds (100, 55, 150, 25);
txtDate.addKeyListener (new KeyAdapter () {
public void keyTyped (KeyEvent ke) {
char c = ke.getKeyChar ();
if (! ((Character.isDigit (c)) || (c == KeyEvent.VK_BACK_SPACE))) {
getToolkit().beep ();
ke.consume ();
}
}
}
);
txtBooks = new JTextField();
txtBooks.setBounds(100,90,150,25);
txtBooks.addKeyListener (new KeyAdapter () {
public void keyTyped (KeyEvent ke) {
char c = ke.getKeyChar ();
if (! ((Character.isDigit (c)) || (c == KeyEvent.VK_BACK_SPACE))) {
getToolkit().beep ();
ke.consume ();
}
}
}
);
btnOk = new JButton ("OK");
btnOk.setBounds (20, 123, 100, 25);
btnOk.addActionListener (this);
btnCancel = new JButton ("Cancel");
btnCancel.setBounds (150, 123, 100, 25);
btnCancel.addActionListener (this);
pNew.setLayout (null);
pNew.add (lbUser);
pNew.add (lbDate);
pNew.add (lbBooks);
pNew.add (txtUser);
```

```
pNew.add (txtDate);
pNew.add (txtBooks);
pNew.add (btnOk);
pNew.add (btnCancel);
getContentPane().add (pNew);
try {
st = con.createStatement ();
}
catch (SQLException sqlex) {
JOptionPane.showMessageDialog (null, "A Problem Occurs While Loading the Form.");
dispose ();
}
setVisible (true);
}
public void actionPerformed (ActionEvent ae) {
Object obj = ae.getSource();
if (obj == btnOk) {
if (txtUser.getText().equals ("")) {
txtUser.requestFocus();
JOptionPane.showMessageDialog (this, "Username not Provided.");
}
else {
try {
String id= txtUser.getText();
String q = "SELECT CName FROM MeCat ";
ResultSet rs = st.executeQuery (q);
int fl=0;
while(rs.next())
{
String memberNo = rs.getString ("CName");
if(id.equals(memberNo))
{
JOptionPane.showMessageDialog(this,"Already existing Category");
txtUser.setText("");
txtUser.requestFocus();
```

```
fl=1;
break;
}
}
rs.close();
int num=0;
try{
rs= st.executeQuery("Select * From MeCat");
while(rs.next())
{
num++;
}
num++;
rs.close();
}
catch(Exception e)
{
JOptionPane.showMessageDialog (this, "Problem while Creating excep1.");
}
if(fl==0){
int result = st.executeUpdate ("Insert into MeCat Values(" + num + ", '" + txtUser.getText() + "' ," +
Integer.parseInt(txtBooks.getText()) + " , " + Integer.parseInt(txtDate.getText())+ " )" );//Running Query.
if (result == 1) {
JOptionPane.showMessageDialog (this, "New Category Created.");
txtUser.setText ("");
txtUser.requestFocus ();
}
else {
JOptionPane.showMessageDialog (this, "Problem while Creating. ");
txtUser.setText ("");
txtUser.requestFocus ();
}
}
}
catch (SQLException sqlex) {
JOptionPane.showMessageDialog (this, "Problem while Creating excep.");
```

```
}
}


}
if (obj == btnCancel) {
setVisible (false);
dispose();
}
}
}
```

Addmember.java

```
import java.awt.*;
import java.awt.event.*;
import java.util.Calendar;
import javax.swing.*;
import java.sql.*;
import java.util.*;

public class AddMember extends JInternalFrame implements ActionListener, FocusListener {
JPanel pMember = new JPanel ();
JLabel lbMemberId, lbMemberName, lbMemberpwd, lbEntryDate,lbCategory;
JTextField txtMemberId, txtMemberName, txtMemberpwd,txtMemberdate;
JButton btnOk, btnCancel;
JComboBox cboMemCategory;
Statement st;
long id = 0;
String[] cn= new String[100];
int id1,im,iy,vd,vm,vy,i;
public AddMember (Connection con) {
super ("Add New Member", false, true, false, true);
setSize (355, 250);
lbMemberId = new JLabel ("Member Id:");
lbMemberId.setForeground (Color.black);
```

```
lbMemberId.setBounds (15, 15, 100, 20);
lbMemberName = new JLabel ("Member Name:");
lbMemberName.setForeground (Color.black);
lbMemberName.setBounds (15, 45, 100, 20);
lbMemberpwd = new JLabel ("Member Pwd:");
lbMemberpwd.setForeground (Color.black);
lbMemberpwd.setBounds (15, 75, 110, 20);
lbEntryDate = new JLabel ("Entry Date:");
lbEntryDate.setForeground (Color.black);
lbEntryDate.setBounds (15, 105, 100, 20);
lbCategory = new JLabel ("Category:");
lbCategory.setForeground(Color.BLACK);
lbCategory.setBounds(15,135,100,20);

txtMemberId = new JTextField ();
txtMemberId.setHorizontalAlignment (JTextField.RIGHT);
txtMemberId.addFocusListener (this);
txtMemberId.setBounds (125, 15, 205, 25);
txtMemberName = new JTextField ();
txtMemberName.setBounds (125, 45, 205, 25);
txtMemberpwd = new JTextField ();
txtMemberpwd.setBounds (125, 75, 205, 25);
txtMemberdate=new JTextField();
txtMemberdate.setBounds(125,105,205,25);
txtMemberdate.setEditable(false);
cboMemCategory= new JComboBox();
cboMemCategory.setBounds(125,135,100,20);
GregorianCalendar gcal=new GregorianCalendar();
id= gcal.get(Calendar.DATE);
im=(int)gcal.get(Calendar.MONTH)+1;
iy=gcal.get(Calendar.YEAR);
String idate=id+"/"+im+"/"+iy;
txtMemberdate.setText(idate);
btnOk = new JButton ("OK");
btnOk.setBounds (30, 165, 125, 25);
```

```
btnOk.addActionListener (this);
btnCancel = new JButton ("Cancel");
btnCancel.setBounds (190, 165, 125, 25);
btnCancel.addActionListener (this);
txtMemberId.addKeyListener (new KeyAdapter () {
public void keyTyped (KeyEvent ke) {
char c = ke.getKeyChar ();
if (! ((Character.isDigit (c)) || (c == KeyEvent.VK_BACK_SPACE))) {
getToolkit().beep ();
ke.consume ();
}
}
}
);
txtMemberName.addKeyListener(new KeyAdapter() {
public void keyTyped(KeyEvent ke) {
char c = ke.getKeyChar();
if (! ((Character.isLetter(c)) || (c == KeyEvent.VK_BACK_SPACE)||(c == KeyEvent.VK_SPACE))) {
getToolkit().beep();
ke.consume();
}
}
}
 );
pMember.setLayout (null);
pMember.add (lbMemberId);
pMember.add (lbMemberName);
pMember.add (lbMemberpwd);
pMember.add (lbEntryDate);
pMember.add (txtMemberId);
pMember.add (txtMemberName);
pMember.add (txtMemberpwd);
pMember.add(txtMemberdate);
pMember.add (btnOk);
pMember.add (btnCancel);
```

```
pMember.add (lbCategory);
pMember.add (cboMemCategory);
getContentPane().add (pMember, BorderLayout.CENTER);
int j;
try {
i=0;
st = con.createStatement ();
ResultSet rs=st.executeQuery("Select * from MeCat");
while(rs.next())
{
cn[i]=rs.getString(2);
i++;
}
for(j=0;j<i;j++)
{
cboMemCategory.addItem(cn[j]);
}
cboMemCategory.addActionListener(this);
cboMemCategory.setSelectedItem(cn[0]);
rs.close();
}
catch (SQLException sqlex) {
JOptionPane.showMessageDialog (null, "A Problem Occurs While Loading Form.");
dispose ();
}
setVisible (true);
}
public void actionPerformed (ActionEvent ae) {
Object obj = ae.getSource();
if (obj == btnOk) {
if (txtMemberId.getText().equals ("")) {
JOptionPane.showMessageDialog (this, "Member's Id not Provided.");
txtMemberId.requestFocus ();
}
else if (txtMemberName.getText().equals ("")) {
```

```
JOptionPane.showMessageDialog (this, "Member's Name not Provided.");

txtMemberName.requestFocus ();

}

else if (txtMemberpwd.getText().equals ("")) {

JOptionPane.showMessageDialog (this, "Member's Password not Provided.");

txtMemberpwd.requestFocus ();

}

else {

try {

int mtype=cboMemCategory.getSelectedIndex()+1;

String q = "INSERT INTO Members" + " VALUES (" + id + ", '" + txtMemberpwd.getText() + "', '" + txtMemberName.getText() + "',
'" + txtMemberdate.getText() + "',"+ 0 + "," + 0 + "," + mtype + ")";

int result = st.executeUpdate (q);

if (result == 1) {

JOptionPane.showMessageDialog (this, "Record has been Saved.");

txtClear ();

}

else {

JOptionPane.showMessageDialog (this, "Problem while Saving the Record.");

}

}

catch (SQLException sqlex) {JOptionPane.showMessageDialog(this,"Error!!"); }

}

}

if (obj == btnCancel) {

setVisible (false);

dispose();

}

}

public void focusGained (FocusEvent fe) { }

public void focusLost (FocusEvent fe) {

if (txtMemberId.getText().equals ("")) {

}

else {

id = Integer.parseInt (txtMemberId.getText ());

long memberNo;
```

```
boolean found = false;
try {
String q = "SELECT * FROM Members WHERE id = " + id + "";
ResultSet rs = st.executeQuery (q);
rs.next ();
memberNo = rs.getLong ("id");
if (memberNo == id) {
found = true;
txtClear ();
JOptionPane.showMessageDialog (this, id + " is already assigned.");
}
else
{
found = false;
}
}
catch (SQLException sqlex) { }
}
}
private void txtClear () {
txtMemberId.setText ("");
txtMemberName.setText ("");
txtMemberpwd.setText ("");
txtMemberId.requestFocus ();
}
}
```

Dates.java

```
public class Dates
{
int m,d,y;
public Dates(int month, int day, int year)
{
m=month;
```

```
d=day;
y=year;
}

public int getMonth()
{ return m; }

public int getDay()
{ return d; }

public int getYear()
{ return y; }

public void setMonth(int month)
{ m=month;  }

public void setDay(int day)
{ d=day;    }

public void setYear(int year)
{ y=year;   }

public String toString()
{
String s = m + "/" + d + "/" + y;
return s;
}

public long toLong()
{
long days=0;

switch(m)
{
case 12: days+=30;
```

```
case 11: days+=31;
case 10: days+=30;
case 9:  days+=31;
case 8:  days+=31;
case 7:  days+=30;
case 6:  days+=31;
case 5:  days+=30;
case 4:  days+=31;
case 3:  days+= isLeapYear(y) ? 29 : 28;
case 2:  days+=31;
case 1:  days+=d-1;
}

if(y!=1900) {
int inc=(1900-y)/Math.abs(1900-y);
for(int i=y; i!=1900; i+=inc)
days += (isLeapYear(i) ? 366 : 365);
}

return days;
}

private boolean isLeapYear(int y)
{
 if((y%100)==0) return (y%400)==0;
 else return (y%4)==0;
 }

public long getDifference(Dates date)
{ return Math.abs(toLong()-date.toLong()); }

}
```

DeleteBook.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;
import java.util.*;
public class DeleteBook extends JInternalFrame implements ActionListener, FocusListener {
JPanel pBook = new JPanel ();
JLabel lbBookId, lbBookName, lbBookAuthor;
JTextField txtBookId, txtBookName, txtBookAuthor;
JButton btnDel, btnCancel;
Statement st;
ResultSet rs;
private long id = 0,bisued;
public DeleteBook (Connection con) {
super ("Delete Book", false, true, false, true);
setSize (325, 250);
lbBookId = new JLabel ("Book Id:");
lbBookId.setForeground (Color.black);
lbBookId.setBounds (15, 15, 100, 20);
lbBookName = new JLabel ("Book Name:");
lbBookName.setForeground (Color.black);
lbBookName.setBounds (15, 45, 100, 20);
lbBookAuthor = new JLabel ("Book Author:");
lbBookAuthor.setForeground (Color.black);
lbBookAuthor.setBounds (15, 75, 100, 20);
txtBookId = new JTextField ();
txtBookId.setHorizontalAlignment (JTextField.RIGHT);
txtBookId.addFocusListener (this);
txtBookId.setBounds (120, 15, 175, 25);
txtBookName = new JTextField ();
txtBookName.setEnabled (false);
txtBookName.setBounds (120, 45, 175, 25);
txtBookAuthor = new JTextField ();
txtBookAuthor.setEnabled (false);
txtBookAuthor.setBounds (120, 75, 175, 25);
```

```
btnDel = new JButton ("Delete Book");
btnDel.setBounds (25, 175, 125, 25);
btnDel.addActionListener (this);
btnCancel = new JButton ("Cancel");
btnCancel.setBounds (165, 175, 125, 25);
btnCancel.addActionListener (this);
txtBookId.addKeyListener (new KeyAdapter () {
public void keyTyped (KeyEvent ke) {
char c = ke.getKeyChar ();
if (! ((Character.isDigit (c)) || (c == KeyEvent.VK_BACK_SPACE))) {
getToolkit().beep ();
ke.consume ();
}
}
}
);

pBook.setLayout (null);
pBook.add (lbBookId);
pBook.add (lbBookName);
pBook.add (lbBookAuthor);
pBook.add (txtBookId);
pBook.add (txtBookName);
pBook.add (txtBookAuthor);
pBook.add (btnDel);
pBook.add (btnCancel);
getContentPane().add (pBook, BorderLayout.CENTER);
try {
st = con.createStatement ();
}
catch (SQLException sqlex) {
JOptionPane.showMessageDialog (null, "A Problem Occurs While Loading the Form.");
dispose ();
}
setVisible (true);
```

```
}

public void actionPerformed (ActionEvent ae) {
Object obj = ae.getSource();
if (obj == btnDel) {
if (txtBookId.getText().equals ("")) {
JOptionPane.showMessageDialog (this, "Book's Id not Provided.");
txtBookId.requestFocus ();
}
else if(bisued!=0)
{
txtClear();
JOptionPane.showMessageDialog(this,"Book held by a member");
}
else
{

int reply = JOptionPane.showConfirmDialog (this, "Are you really want to Delete\nthe " + txtBookName.getText () + "
Record?","LibrarySystem - Delete Book", JOptionPane.YES_NO_OPTION, JOptionPane.PLAIN_MESSAGE);

if (reply == JOptionPane.YES_OPTION) {
try {
String q = "DELETE FROM Books WHERE BId = " + id + "";
txtClear ();
JOptionPane.showMessageDialog (this, "Book Deleted.");
ResultSet rs = st.executeQuery (q);
}
catch (SQLException sqlex) { }
}

else if (reply == JOptionPane.NO_OPTION) { }
}
}
if (obj == btnCancel) {
setVisible (false);
dispose();
```

```
}
}
public void focusGained (FocusEvent fe) { }
public void focusLost (FocusEvent fe) {
if (txtBookId.getText().equals ("")) {
}
else {
id = Integer.parseInt (txtBookId.getText ());
long bookNo;
boolean found = false;
try {
String q = "SELECT * FROM Books WHERE BId = " + id + "";
ResultSet rs = st.executeQuery (q);
rs.next ();
bookNo = rs.getLong ("BId");
bisued=rs.getLong("Mid");
if (bookNo == id) {
found = true;
txtBookId.setText ("" + id);
txtBookName.setText ("" + rs.getString ("BName"));
txtBookAuthor.setText ("" + rs.getString ("BAuthor"));
}
else {
found = false;
}
}
catch (SQLException sqlex) {
if (found == false) {
txtClear ();
JOptionPane.showMessageDialog (this, "Record not Found.");
}
}
}
}
private void txtClear () {
```

```
txtBookId.setText ("");
txtBookName.setText ("");
txtBookAuthor.setText ("");
txtBookId.requestFocus();
}
}
```

DeleteMember.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;

public class DeleteMember extends JInternalFrame implements ActionListener, FocusListener {
private JPanel pMember = new JPanel ();
private JLabel lbMemberId, lbMemberName,lbMemberCat;
private JTextField txtMemberId, txtMemberName,txtCat;
private JButton btnDel, btnCancel;
private int due;
private Statement st;                    //Statement for Getting the Required Table.
private ResultSet rs;                    //For Getting the Records From Table.
private long id = 0,heldBooks;                    //To Hold the MemberId.
//Constructor of Class.
public DeleteMember (Connection con) {
//super (Title, Resizable, Closable, Maximizable, Iconifiable)
super ("Delete Member", false, true, false, true);
setSize (350, 222);
//Setting the Form's Labels.
lbMemberId = new JLabel ("Member Id:");
lbMemberId.setForeground (Color.black);
lbMemberId.setBounds (15, 15, 100, 20);
lbMemberName = new JLabel ("Member Name:");
lbMemberName.setForeground (Color.black);
lbMemberName.setBounds (15, 45, 100, 20);
```

```
lbMemberCat = new JLabel ("Category");
lbMemberCat.setForeground (Color.black);
lbMemberCat.setBounds (15, 75, 110, 20);
txtMemberId = new JTextField ();
txtMemberId .setHorizontalAlignment (JTextField.RIGHT);
txtMemberId.addFocusListener (this);
txtMemberId .setBounds (125, 15, 200, 25);
txtMemberName = new JTextField ();
txtMemberName.setEnabled (false);
txtMemberName.setBounds (125, 45, 200, 25);
txtCat = new JTextField ();
txtCat.setEnabled (false);
txtCat.setBounds (125, 75, 200, 25);
btnDel = new JButton ("Delete Member");
btnDel.setBounds (30, 145, 125, 25);
btnDel.addActionListener (this);
btnCancel = new JButton ("Cancel");
btnCancel.setBounds (185, 145, 125, 25);
btnCancel.addActionListener (this);
//Registering the KeyListener to Restrict user to type only Numeric in Numeric Boxes.
txtMemberId.addKeyListener (new KeyAdapter () {
public void keyTyped (KeyEvent ke) {
char c = ke.getKeyChar ();
if (! ((Character.isDigit (c)) || (c == KeyEvent.VK_BACK_SPACE))) {
getToolkit().beep ();
ke.consume ();
}
}
}
);
//Adding All the Controls in Panel.
pMember.setLayout (null);
pMember.add (lbMemberId);
pMember.add (lbMemberName);
pMember.add(lbMemberCat);
```

```
pMember.add(txtCat);
pMember.add (txtMemberId);
pMember.add (txtMemberName);
pMember.add (btnDel);
pMember.add (btnCancel);
//Adding Panel to Form.
getContentPane().add (pMember, BorderLayout.CENTER);
try {
st = con.createStatement ();              //Creating Statement Object.
}
catch (SQLException sqlex) {                                    //If Problem then Show the User a Message.
JOptionPane.showMessageDialog (null, "A Problem Occurs While Loading the Form.");
dispose ();                                                     //Closing the Form.
}
setVisible (true);
}
public void actionPerformed (ActionEvent ae) {
Object obj = ae.getSource();
if (obj == btnDel) {                        //If Delete Button Pressed.
if (txtMemberId.getText().equals ("")) {
JOptionPane.showMessageDialog (this, "Member's Id not Provided.");
txtMemberId.requestFocus ();
}
else if(heldBooks!=0)
{
JOptionPane.showMessageDialog(this,"Member Holding Books..Can't Delete");
txtClear();
}
else if(due!=0)
{
JOptionPane.showMessageDialog(this,"Member Holding Books..Can't Delete");
txtClear();
}
else
{
```

```
int reply = JOptionPane.showConfirmDialog (this, "Do you really want to Delete\nthe " + txtMemberName.getText () + "
Record?","LibrarySystem - Delete Member", JOptionPane.YES_NO_OPTION, JOptionPane.PLAIN_MESSAGE);


//Check the User Selection.
if (reply == JOptionPane.YES_OPTION) {                    //If User's Choice Yes then.
try {       //DELETE Query to Delete the Record from Table.
String q = "DELETE FROM Members WHERE id = " + id + "";
txtClear ();                                    //Clearing the TextFields.
JOptionPane.showMessageDialog (this, "Record has been Deleted.");
ResultSet rs = st.executeQuery (q);  //Executing the Query.
}
catch (SQLException sqlex) {System.out.println("problem"); }
}
//If User's Choice No then Do Nothing Return to Program.
else if (reply == JOptionPane.NO_OPTION) { }
}
}
if (obj == btnCancel) {       //If Cancel Button Pressed Unload the From.
setVisible (false);
dispose();
}
}


//OverRidding the FocusListener Class Function.
public void focusGained (FocusEvent fe) { }
public void focusLost (FocusEvent fe) {
if (txtMemberId.getText().equals ("")) {         //If TextField is Empty.
}
else {
id = Integer.parseInt (txtMemberId.getText ());//Converting String to Numeric.
long memberNo,memtype;                    //Use for Comparing the Member's Id.
boolean found = false;                              //To Confirm the Member's Id Existance.
try {       //SELECT Query to Retrieved the Record.
String q = "SELECT * FROM Members WHERE id = " + id + "";
ResultSet rs = st.executeQuery (q);  //Executing the Query.
rs.next ();                                     //Moving towards the Record.
```

```
memberNo = rs.getLong ("id");        //Storing the Record.
heldBooks=rs.getLong("Bcnt");
due=rs.getInt(6);
memtype=rs.getLong("Mcat");
if (memberNo == id) {        //If Record Found then Display Records.
found = true;
txtMemberId.setText ("" + id);
txtMemberName.setText ("" + rs.getString ("MName"));
ResultSet rs1=st.executeQuery("Select * From MeCat where Mcat="+memtype+"");
rs1.next();
txtCat.setText(""+rs1.getString("CName"));
}
else {
found = false;
}
}
catch (SQLException sqlex) {
if (found == false) {
txtClear ();                        //Clearing the TextFields.
JOptionPane.showMessageDialog (this, "Record not Found.");
}
}
}
}
//Function Use to Clear All the TextFields of Form.
private void txtClear () {
txtMemberId.setText ("");
txtMemberName.setText ("");
txtCat.setText("");
txtMemberId.requestFocus ();
}
}
```

IssueBook.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.Calendar;
import java.sql.*;
import java.util.*;

public class IssueBook extends JInternalFrame implements ActionListener, FocusListener {
private JPanel pBook = new JPanel ();
private JLabel lbBookId, lbBookName, lbBookAuthor, lbBookCategory, lbMemberId, lbMemberName,lbDate1,lbDate2;
private JTextField txtBookId, txtBookName, txtBookAuthor, txtBookCategory, txtMemberId, txtMemberName,txtDate1,txtDate2;
private JButton btnOk, btnCancel;
private Statement st;
private long id = 0;
private int memberId = 0;
private int id1,im,iy,vd,vm,vy;
private String idate,vdate;
public IssueBook (Connection con) {
super ("Issue Book", false, true, false, true);
setSize (325, 340);
lbBookId = new JLabel ("Book Id:");
lbBookId.setForeground (Color.black);
lbBookId.setBounds (15, 15, 100, 20);
lbBookName = new JLabel ("Book Name:");
lbBookName.setForeground (Color.black);
lbBookName.setBounds (15, 45, 100, 20);
lbBookAuthor = new JLabel ("Book Author:");
lbBookAuthor.setForeground (Color.black);
lbBookAuthor.setBounds (15, 75, 100, 20);
lbBookCategory = new JLabel ("Book Category:");
lbBookCategory.setForeground (Color.black);
lbBookCategory.setBounds (15, 105, 100, 20);
lbMemberId = new JLabel ("Member Id:");
```

```
lbMemberId.setForeground (Color.black);
lbMemberId.setBounds (15, 135, 100, 20);
lbMemberName = new JLabel ("Member Name:");
lbMemberName.setForeground (Color.black);
lbMemberName.setBounds (15, 165, 100, 20);
lbDate1 = new JLabel ("Issue Date:");
lbDate1.setForeground (Color.black);
lbDate1.setBounds (15, 195, 100, 20);
lbDate2 = new JLabel ("Return Date:");
lbDate2.setForeground (Color.black);
lbDate2.setBounds (15, 225, 100, 20);
txtBookId = new JTextField ();
txtBookId.setHorizontalAlignment (JTextField.RIGHT);
txtBookId.addFocusListener (this);
txtBookId.setBounds (120, 15, 175, 25);
txtBookName = new JTextField ();
txtBookName.setEnabled (false);
txtBookName.setBounds (120, 45, 175, 25);
txtBookAuthor = new JTextField ();
txtBookAuthor.setEnabled (false);
txtBookAuthor.setBounds (120, 75, 175, 25);
txtBookCategory = new JTextField ();
txtBookCategory.setEnabled (false);
txtBookCategory.setBounds (120, 105, 175, 25);
txtMemberId = new JTextField ();
txtMemberId.setHorizontalAlignment (JTextField.RIGHT);
txtMemberId.addFocusListener (this);
txtMemberId.setBounds (120, 135, 175, 25);
txtMemberName = new JTextField ();
txtMemberName.setEnabled (false);
txtMemberName.setBounds (120, 165, 175, 25);
txtDate1 = new JTextField ();
txtDate1.setEnabled (false);
txtDate1.setBounds (120, 195, 175, 25);
txtDate1.setEditable(false);
```

```
txtDate2 = new JTextField ();
txtDate2.setEnabled (false);
txtDate2.setBounds (120, 225, 175, 25);
txtDate2.setEditable(false);
btnOk = new JButton ("OK");
btnOk.setBounds (50, 260, 100, 25);
btnOk.addActionListener (this);
btnCancel = new JButton ("Cancel");
btnCancel.setBounds (170, 260, 100, 25);
btnCancel.addActionListener (this);
txtBookId.addKeyListener (new KeyAdapter () {
public void keyTyped (KeyEvent ke) {
char c = ke.getKeyChar ();
if (! ((Character.isDigit (c)) || (c == KeyEvent.VK_BACK_SPACE))) {
getToolkit().beep ();
ke.consume ();
}
}
}
);

txtMemberId.addKeyListener (new KeyAdapter () {
public void keyTyped (KeyEvent ke) {
char c = ke.getKeyChar ();
if (! ((Character.isDigit (c)) || (c == KeyEvent.VK_BACK_SPACE))) {
getToolkit().beep ();
ke.consume ();
}
}
}
);

pBook.setLayout (null);
pBook.add (lbBookId);
pBook.add (lbBookName);
```

```
pBook.add (lbBookAuthor);
pBook.add (lbBookCategory);
pBook.add (lbMemberId);
pBook.add (lbMemberName);
pBook.add (txtBookId);
pBook.add (txtBookName);
pBook.add (txtBookAuthor);
pBook.add (txtBookCategory);
pBook.add (txtMemberId);
pBook.add (txtMemberName);
pBook.add (btnOk);
pBook.add (btnCancel);
pBook.add (txtDate1);
pBook.add (txtDate2);
pBook.add (lbDate1);
pBook.add (lbDate2);
getContentPane().add (pBook, BorderLayout.CENTER);
try {
st = con.createStatement ();
}
catch (SQLException sqlex) {
JOptionPane.showMessageDialog (null, "A Problem Occurs While Loading Form.");
dispose ();
}
setVisible (true);
}
public void actionPerformed (ActionEvent ae) {
Object obj = ae.getSource();
if (obj == btnOk) {
if (txtBookId.getText().equals ("")) {
JOptionPane.showMessageDialog (this, "Book's Id not Provided.");
txtBookId.requestFocus ();
}
else if (txtMemberId.getText().equals ("")) {
JOptionPane.showMessageDialog (this, "Member's Id not Provided.");
```

```
txtMemberId.requestFocus ();
}
else {
try  {
int i1= Integer.parseInt(txtMemberId.getText());
ResultSet rs = st.executeQuery("Select * from Members where id="+ i1 +"");
rs.next();
int bc=rs.getInt("Bcnt");
bc++;
int bid1=Integer.parseInt(txtBookId.getText());
int result = st.executeUpdate ("Update Members SET Bcnt="+ bc +" WHERE id="+ i1 +"");
if (result == 1) {
txtClear ();
}
else {
txtClear ();
JOptionPane.showMessageDialog (this, "Problem while Saving the Record.");
}
System.out.println("came 1");
result = st.executeUpdate("Update Books SET Mid= "+ i1 + ", BIssue = '"+ idate +"', BReturn = '"+vdate+"' where Bid="+bid1+"");
System.out.println("came 2");
if (result == 1) {
txtClear ();
JOptionPane.showMessageDialog (this, "Record has been Saved.");
}
else {
txtClear ();
JOptionPane.showMessageDialog (this, "Problem while Saving the Record.");
}
}
catch (SQLException sqlex) {JOptionPane.showMessageDialog (this, "Problem"); }
}
}
if (obj == btnCancel) {
setVisible (false);
```

```
dispose();
}
}
public void focusGained (FocusEvent fe) { }
public void focusLost (FocusEvent fe) {
Object obj = fe.getSource ();

if (obj == txtBookId) {
if (txtBookId.getText().equals ("")) {
}
else {
id = Integer.parseInt (txtBookId.getText ());
long bookNo;
boolean found = false;
try {
String q = "SELECT * FROM Books WHERE BId = " + id + "";
ResultSet rs = st.executeQuery (q);
rs.next ();
bookNo = rs.getLong ("BId");
int mid=rs.getInt("Mid");
int bref=rs.getInt("BRef");
if(bref==1)
{
txtClear();
JOptionPane.showMessageDialog (this, "Ref Book Can't Be Issued.");
}
if(mid!=0)
{
txtClear();
JOptionPane.showMessageDialog(this,"Book Already Issued");
}
if (bookNo == id) {
found = true;
txtBookId.setText ("" + id);
txtBookName.setText ("" + rs.getString ("BName"));
```

```
txtBookAuthor.setText ("" + rs.getString ("BAuthor"));
txtBookCategory.setText ("" + rs.getString ("BCat"));
}
else {
found = false;
}
}
catch (SQLException sqlex) {
if (found == false) {
txtBookId.requestFocus ();
txtBookId.setText ("");
txtBookName.setText ("");
txtBookAuthor.setText ("");
txtBookCategory.setText ("");
JOptionPane.showMessageDialog (this, "Record not Found.");
}
}
}
}
else if (obj == txtMemberId) {
if (txtMemberId.getText().equals ("")) {
}
else {
memberId = Integer.parseInt (txtMemberId.getText ());
int memberNo,memberDays,memberBooks,memberCat,heldBooks;
boolean find = false;
try {
String q = "SELECT * FROM Members WHERE id = " + memberId + "";
ResultSet rs = st.executeQuery (q);
rs.next ();
memberNo = rs.getInt ("id");
if (memberNo == memberId) {
find = true;
memberCat=rs.getInt("MCat");
heldBooks=rs.getInt("Bcnt");
```

```
txtMemberName.setText ("" + rs.getString ("MName"));
rs.close();
ResultSet rs1= st.executeQuery("Select * from Mecat where MCat = " + memberCat + "" );
rs1.next();
memberBooks=rs1.getInt("Blmt");
memberDays=rs1.getInt("Dlmt");
if(heldBooks==memberBooks)
{
txtClear();
JOptionPane.showMessageDialog (this, "Book Limit Reached");
dispose();
}

GregorianCalendar gcal=new GregorianCalendar();
 id1= gcal.get(Calendar.DATE);
 im=(int)gcal.get(Calendar.MONTH)+1;
 iy=gcal.get(Calendar.YEAR);
 vd=id1+memberDays;
 vm=im;
 vy=iy;
 String xx,yy,zz;
 if(id1<10) {
 xx="0"+id1;
 }
 else
{
 xx = ""+id1;
 }
if(im<10) {
yy="0"+im;
}
else
{
 yy = ""+im;
 }
```

```
 idate=xx+"/"+yy+"/"+iy;
 while(vd>31) {
 if(im==1||im==3||im==5||im==7||im==8||im==10||im==12)
{
if(vd>31){
im=im+1;
 vd=vd-31;
  if(im>12){
  im=im-12;
  iy=iy+1;
  }}}

 if(im==4||im==6||im==9||im==11){
 if(vd>30){
 im=im+1;
vd=vd-30;
if(im>12){
im=im-12;
 iy=iy+1;}
 }}
if(im==2){
if(vd>28){
im=im+1;
vd=vd-28;
if(im>12){
im=im-12;
iy=iy+1;
}}}
}
vdate = vd+"/"+im+"/"+iy;
txtMemberId.setText ("" + memberId);
txtDate1.setText(idate);
txtDate2.setText(vdate);
}
else {
```

```
find = false;
}
}
catch (SQLException sqlex) {
if (find == false) {
txtClear ();
JOptionPane.showMessageDialog (this, "Record not Found.");
}
}
}
}
}
private void txtClear () {
txtBookId.setText ("");
txtBookName.setText ("");
txtBookAuthor.setText ("");
txtBookCategory.setText ("");
txtMemberId.setText ("");
txtMemberName.setText ("");
txtBookId.requestFocus ();
}
}
```

LibrarySystem.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;
import java.util.*;
import java.text.*;
import java.io.*;

public class LibrarySystem extends JFrame implements ActionListener
{
```

```
private JDesktopPane desktop = new JDesktopPane ();
JMenuBar bar;
JMenu mnuFile, mnuEdit;
JMenuItem newBook,newMember, printBook,  printIssueBook;
JMenuItem issueBook, returnBook, delBook, findBook;
private    JToolBar toolBar;
private    JButton btnNewBook,  btnIssue, btnReturn, btnPrintIssue, btnDelBook,btnFindBook;
private JPanel statusBar = new JPanel ();
Connection con;
Statement st;
String userName;
public LibrarySystem (int type,int user, Connection conn)
{
super ("Library Management System.");
setIconImage (getToolkit().getImage ("Images/Warehouse.png"));
setSize (700, 550);
setLocation((Toolkit.getDefaultToolkit().getScreenSize().width  - getWidth()) / 2,
(Toolkit.getDefaultToolkit().getScreenSize().height - getHeight()) / 2);
addWindowListener (new WindowAdapter () {
public void windowClosing (WindowEvent we) {
//quitApp ();
}
}
);
bar = new JMenuBar ();
setJMenuBar (bar);
mnuFile = new JMenu ("File");
mnuFile.setMnemonic ((int)'E');
mnuEdit = new JMenu ("Edit");
mnuEdit.setMnemonic ((int)'E');
newBook = new JMenuItem ("Add New Book");
newBook.setAccelerator (KeyStroke.getKeyStroke(KeyEvent.VK_N, Event.CTRL_MASK));
newBook.setMnemonic ((int)'N');
newBook.addActionListener (this);
newMember = new JMenuItem ("Add New Member");
```

```
newMember.setAccelerator (KeyStroke.getKeyStroke(KeyEvent.VK_M, Event.CTRL_MASK));
newMember.setMnemonic ((int)'M');
newMember.addActionListener (this);
issueBook = new JMenuItem ("Issue Book");
issueBook.setAccelerator (KeyStroke.getKeyStroke(KeyEvent.VK_I, Event.CTRL_MASK));
issueBook.setMnemonic ((int)'I');
issueBook.addActionListener (this);
returnBook = new JMenuItem ("Return Book");
returnBook.setAccelerator (KeyStroke.getKeyStroke(KeyEvent.VK_R, Event.CTRL_MASK));
returnBook.setMnemonic ((int)'R');
returnBook.addActionListener (this);
delBook = new JMenuItem ("Delete Book");
delBook.setAccelerator (KeyStroke.getKeyStroke(KeyEvent.VK_D, Event.CTRL_MASK));
delBook.setMnemonic ((int)'D');
delBook.addActionListener (this);
findBook = new JMenuItem ("Search Book");
findBook.setAccelerator (KeyStroke.getKeyStroke(KeyEvent.VK_F, Event.CTRL_MASK));
findBook.setMnemonic ((int)'F');
findBook.addActionListener (this);
mnuFile.add (newBook);
mnuFile.add (newMember);
mnuEdit.add (issueBook);
mnuEdit.add (returnBook);
mnuEdit.addSeparator ();
mnuEdit.add (delBook);
mnuEdit.addSeparator ();
mnuEdit.add (findBook);
bar.add (mnuFile);
bar.add (mnuEdit);
btnNewBook = new JButton (new ImageIcon ("Images/NotePad.gif"));
btnNewBook.setToolTipText ("Add New Book");
btnIssue = new JButton (new ImageIcon ("Images/Film.gif"));
btnIssue.setToolTipText ("Issue Book");
btnReturn = new JButton (new ImageIcon ("Images/Backup.gif"));
btnReturn.setToolTipText ("Return Book");
```

```
btnDelBook = new JButton (new ImageIcon ("Images/Recycle.gif"));
btnDelBook.setToolTipText ("Delete Book");
btnFindBook = new JButton (new ImageIcon ("Images/Mirror.gif"));
btnFindBook.setToolTipText ("Search Book");
btnFindBook.addActionListener (this);
toolBar = new JToolBar ();
toolBar.add (btnNewBook);
toolBar.addSeparator ();
toolBar.add (btnIssue);
toolBar.add (btnReturn);
toolBar.addSeparator ();
toolBar.add (btnDelBook);
toolBar.addSeparator ();
toolBar.add (btnFindBook);
if(type==1)
userName="Admin";
else if(type==2)
{
}
else if(type==3)
{
}

//Setting the Contents of Programs.
getContentPane().add (toolBar, BorderLayout.NORTH);
getContentPane().add (desktop, BorderLayout.CENTER);
getContentPane().add (statusBar, BorderLayout.SOUTH);
//Getting the Database.
con = conn;
setVisible (true);
}
public void actionPerformed (ActionEvent ae) {
Object obj = ae.getSource();
if (obj == newBook ) {
```

```
boolean b = openChildWindow ("Add New Book");
if (b == false) {
AddBook adBook = new AddBook (con);
desktop.add (adBook);
adBook.show ();
}
}
else if (obj == newMember )
 {
boolean b = openChildWindow ("Add New Member");
if (b == false) {
AddMember adMember = new AddMember (con);
desktop.add (adMember);
adMember.show ();
}
}
else if (obj == issueBook )
 {
boolean b = openChildWindow ("Issue Book");
if (b == false) {
IssueBook isBook = new IssueBook (con);
desktop.add (isBook);
isBook.show ();
}
}
else if (obj == returnBook)
{
boolean b = openChildWindow ("Return Book");
if (b == false) {
ReturnBook rtBook = new ReturnBook (con);
desktop.add (rtBook);
rtBook.show ();
}
}
```

```
else if (obj == delBook)
{
boolean b = openChildWindow ("Delete Book");
if (b == false)
{
DeleteBook dlBook = new DeleteBook (con);
desktop.add (dlBook);
dlBook.show ();
}
}
else if (obj == findBook )
{
boolean b = openChildWindow ("Search Books");
if (b == false) {
SearchBook srBook = new SearchBook (con);
desktop.add (srBook);
srBook.show ();
}
}
}
private boolean openChildWindow (String title)
{
JInternalFrame[] childs = desktop.getAllFrames ();
for (int i = 0; i < childs.length; i++) {
if (childs[i].getTitle().equalsIgnoreCase (title))
{
childs[i].show ();
return true;
}
}
return false;
}
}
```

ReturnBook.java

```
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.Date;
import java.util.Calendar;
import java.util.*;
import java.text.SimpleDateFormat;
import java.sql.*;
public class ReturnBook extends JInternalFrame implements ActionListener, FocusListener {
private JPanel pBook = new JPanel ();
private JLabel lbBookId, lbBookName,lbIssued;
private JTextField txtBookId, txtBookName, txtIssued;
private String urdate;
private JButton btnReturn, btnCancel;
private int id1,im,iy,vd,vm,vy,due;
private Statement st;
private ResultSet rs;
private long id = 0;
private int mid,bc;
public ReturnBook (Connection con) {
super ("Return Book", false, true, false, true);
setSize (325, 250);
lbBookId = new JLabel ("Book Id:");
lbBookId.setForeground (Color.black);
lbBookId.setBounds (15, 15, 100, 20);
lbBookName = new JLabel ("Book Name:");
lbBookName.setForeground (Color.black);
lbBookName.setBounds (15, 45, 100, 20);
lbIssued = new JLabel ("Book Issued To:");
lbIssued.setForeground (Color.black);
lbIssued.setBounds (15, 75, 100, 20);
txtBookId = new JTextField ();
txtBookId.setHorizontalAlignment (JTextField.RIGHT);
```

```
txtBookId.addFocusListener (this);
txtBookId.setBounds (120, 15, 175, 25);
txtBookName = new JTextField ();
txtBookName.setEnabled (false);
txtBookName.setBounds (120, 45, 175, 25);
txtIssued = new JTextField ();
txtIssued.setEnabled (false);
txtIssued.setBounds (120, 75, 175, 25);
btnReturn = new JButton ("Return Book");
btnReturn.setBounds (25, 175, 125, 25);
btnReturn.addActionListener (this);
btnCancel = new JButton ("Cancel");
btnCancel.setBounds (165, 175, 125, 25);
btnCancel.addActionListener (this);
txtBookId.addKeyListener (new KeyAdapter () {
public void keyTyped (KeyEvent ke) {
char c = ke.getKeyChar ();
if (! ((Character.isDigit (c)) || (c == KeyEvent.VK_BACK_SPACE))) {
getToolkit().beep ();
ke.consume ();
}
}
}
);
pBook.setLayout (null);
pBook.add (lbBookId);
pBook.add (lbBookName);
pBook.add (lbIssued);
pBook.add (txtBookId);
pBook.add (txtBookName);
pBook.add (txtIssued);
pBook.add (btnReturn);
pBook.add (btnCancel);
getContentPane().add (pBook, BorderLayout.CENTER);
try {
```

```
st = con.createStatement ();
}
catch (SQLException sqlex) {
JOptionPane.showMessageDialog (null, "A Problem Occurs While Loading the Form.");
dispose ();
}
GregorianCalendar gcal=new GregorianCalendar();
 id1= gcal.get(Calendar.DATE);
 im=(int)gcal.get(Calendar.MONTH)+1;
 iy=gcal.get(Calendar.YEAR);
 String xx,yy,zz;
 if(id1<10) {
 xx="0"+id1;
} else {
 xx = ""+id1;
 }
if(im<10) {
yy="0"+im;
}
 else
{
  yy = ""+im;
 }
 urdate=xx+"/"+yy+"/"+iy;
setVisible (true);
}
public void actionPerformed (ActionEvent ae)
{
Object obj = ae.getSource();
if (obj == btnReturn) {
if (txtBookId.getText().equals (""))
{
JOptionPane.showMessageDialog (this, "Book's Id not Provided.");
txtBookId.requestFocus ();
}
```

```
else {

try {

int rd,rm,ry,urd,urm,ury,x;

long v,v1,fine;

Dates d1,d2;

bc--;

id = Integer.parseInt (txtBookId.getText ());

ResultSet rs = st.executeQuery ("select * from Books WHERE BId ="+id+"");

//Executing the Query.

rs.next();

String ard=rs.getString("BReturn");

System.out.println("came here 1");

rs.close();

String sr=urdate;

StringTokenizer st2 = new StringTokenizer(sr,"/");

urd=Integer.parseInt(st2.nextToken());

urm=Integer.parseInt(st2.nextToken());

ury=Integer.parseInt(st2.nextToken());

d2= new Dates(urm,urd,ury);

StringTokenizer st1 = new StringTokenizer(ard,"/");

rd=Integer.parseInt(st1.nextToken());

rm=Integer.parseInt(st1.nextToken());

ry=Integer.parseInt(st1.nextToken());

d1=new Dates(rm,rd,ry);

v = d1.toLong();

v1 = d2.toLong();

fine=v1-v;

if(fine<=0)

fine=0;

else

{

int reply = JOptionPane.showConfirmDialog (this, "Will you pay the Fine of Rs."+fine+"now","FinePay",
JOptionPane.YES_NO_OPTION, JOptionPane.PLAIN_MESSAGE);

if (reply == JOptionPane.YES_OPTION)

{}

else if (reply == JOptionPane.NO_OPTION)
```

```
{
due+=fine;
}
}
x=st.executeUpdate("Update Books Set Mid ="+0+" WHERE Bid ="+id+"");
x=st.executeUpdate("Update Members Set Bcnt ="+bc+", Mbdues="+due+" WHERE id ="+mid+"");
JOptionPane.showMessageDialog (this, "Book Returned");
txtClear();
catch (SQLException sqlex) {
JOptionPane.showMessageDialog (this, "Problem");
}
}
}
if (obj == btnCancel) {
setVisible (false);
dispose();
}
}
public void focusGained (FocusEvent fe) { }
public void focusLost (FocusEvent fe) {
if (txtBookId.getText().equals ("")) {
}
else {
id = Integer.parseInt (txtBookId.getText ());
long bookNo;
boolean found = false;
try {
ResultSet rs = st.executeQuery ("Select * from Books where BId="+id+"");          //Executing the Query.
rs.next ();
bookNo = rs.getLong ("BId");
if (bookNo == id) {
found = true;
txtBookId.setText ("" + id);
txtBookName.setText ("" + rs.getString ("BName"));
mid=rs.getInt("Mid");
```

```
if(mid==0)
{
JOptionPane.showMessageDialog(this,"Not an Issued Book");
dispose();
}
else
{
ResultSet rs1=st.executeQuery("Select * from Members where id="+mid+"");
rs1.next();
txtIssued.setText ("" + rs1.getString (3));
bc=rs1.getInt("Bcnt");
due=rs1.getInt(6);
}
}
else {
found = false;
}
}
catch (SQLException sqlex) {
if (found == false) {
txtClear ();
JOptionPane.showMessageDialog (this, "Record not Found.");
}
}
}
}

private void txtClear () {
txtBookId.setText ("");
txtBookName.setText ("");
txtIssued.setText ("");
txtBookId.requestFocus ();
}
}
```

SearchBook.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;

public class SearchBook extends JInternalFrame implements ActionListener {
JPanel pBook = new JPanel ();
JLabel lbSearch;
JRadioButton rb1,rb2,rb3,rb4;
JTextField txtSearch;
JButton btnFind, btnCancel;
int flag=0;
Statement st;
String bname,bauthor,bcat,search;
int bref,bmid,bid,rows=0;
JTable table;
JScrollPane jsp;
Object data1[][];
Container c;
public SearchBook (Connection con) {
super ("Search Books", false, true, false, true);
setSize (510, 300);
lbSearch = new JLabel ("Search Field");
lbSearch.setForeground (Color.black);
lbSearch.setBounds (15, 15, 100, 20);
txtSearch = new JTextField ();
txtSearch.setBounds (120, 15, 175, 25);
btnFind = new JButton ("Find Book");
btnFind.setBounds (25, 175, 125, 25);
btnFind.addActionListener (this);
btnCancel = new JButton ("Cancel");
btnCancel.setBounds (165, 175, 125, 25);
```

```
btnCancel.addActionListener (this);
rb1=new JRadioButton("By Title");
rb1.addActionListener(this);
rb1.setBounds (15, 45, 100, 20);
rb2=new JRadioButton("By Author");
rb2.addActionListener(this);
rb2.setBounds (15, 75, 100, 20);
rb3=new JRadioButton("By Category");
rb3.addActionListener(this);
rb3.setBounds (15, 105, 100, 20);
rb4=new JRadioButton("By id");
rb4.addActionListener(this);
rb4.setBounds(15,135,100,20);
pBook.setLayout (null);
pBook.add(lbSearch);
pBook.add(txtSearch);
pBook.add(btnFind);
pBook.add(btnCancel);
ButtonGroup bg=new ButtonGroup();
bg.add(rb1);
bg.add(rb2);
bg.add(rb3);
bg.add(rb4);
pBook.add(rb1);
pBook.add(rb2);
pBook.add(rb3);
pBook.add(rb4);
rb1.setSelected(true);
getContentPane().add (pBook, BorderLayout.CENTER);
c=getContentPane();
try
{
st = con.createStatement ();
}
catch (SQLException sqlex) {
```

```
JOptionPane.showMessageDialog (null, "A Problem Occurs While Loading Form.");
dispose ();
}
setVisible (true);
}
public void actionPerformed (ActionEvent ae) {
Object obj = ae.getSource();
if (obj == btnFind) {
if (txtSearch.getText().equals ("")) {
JOptionPane.showMessageDialog (this, "Search Field not Provided.");
txtSearch.requestFocus ();
}
else
{
String bname1,bauthor1,bcat1;
int num;
boolean found = false;
try {
String q,bavl,bisr;
num=st.executeUpdate("Delete * from BSearch");
ResultSet rs = st.executeQuery ("SELECT * FROM Books ");      //Executing the Query.
search=txtSearch.getText();
search=search.toLowerCase();
while(rs.next())
{
bname=rs.getString(2);
bauthor=rs.getString("BAuthor");
bcat=rs.getString("BCat");
bref=rs.getInt("BRef");
if(bref==1) bisr="Yes";
else bisr="No";
bmid=rs.getInt("Mid");
if(bmid==0) bavl="Available";
else bavl="Issued:"+ bmid;
bid=rs.getInt("BId");
```

```
if(flag==0)
{
bname1=bname.toLowerCase();
if(bname1.equals(search)||(bname1.indexOf(search)!=-1))
{
System.out.println("Came Here2");
num=st.executeUpdate("insert into BSearch values("+bid+", '"+bname+"' , '"+bcat+"' , '"+bauthor+"' , '"+bavl+"', '"+bisr+"')");
rows++;
found=true;
}
}
else if(flag==1)
{
bauthor1=bauthor.toLowerCase();
if(bauthor1.equals(search)||(bauthor1.indexOf(search)!=-1))
{
num=st.executeUpdate("insert into BSearch values("+bid+", '"+bname+"' , '"+bcat+"' , '"+bauthor+"' , '"+bavl+"', '"+bisr+"')");
rows++;
found=true;
}
}
else if(flag==2)
{
bcat1=bcat.toLowerCase();
if(bcat1.equals(search)||(bcat1.indexOf(search)!=-1))
{
num=st.executeUpdate("insert into BSearch values("+bid+", '"+bname+"' , '"+bcat+"' , '"+bauthor+"' , '"+bavl+"', '"+bisr+"')");
rows++;
found=true;
}
}
else if(flag==3)
{
if(bid==Integer.parseInt(txtSearch.getText()))
{
```

```
rows++;
num=st.executeUpdate("insert into BSearch values("+bid+", '"+bname+"' , '"+bcat+"' , '"+bauthor+"' , '"+bavl+"', '"+bisr+"')");
found=true;
}
}
}
}
catch(SQLException sqlex) {
if (found == false) {
JOptionPane.showMessageDialog (this, "Record not Found.");
}
}
try{
data1=new Object[rows][6];
Object[] Colheads={"Book Id","Book Name","Category","Author","Availability","Reference"};
ResultSet rs=st.executeQuery("Select * from BSearch");
for(int i1=0;i1<rows;i1++)
{
rs.next();
for(int j1=0;j1<6;j1++)
{
data1[i1][j1]=rs.getString(j1+1);
}
}
table=new JTable(data1,Colheads);
int v=ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED;
int h=ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED;
System.out.println("hai we came here");
jsp=new JScrollPane(table,v,h);
TableDisp td=new TableDisp(table);



}
catch(Exception sqlex) {
if (found == false) {
```

```
JOptionPane.showMessageDialog (this, "Some prob Found.");
}
}
}
}
if (obj == btnCancel) {
setVisible (false);
dispose();
}
if(obj==rb1)
{
flag=0;
}
if(obj==rb2)
{
flag=1;
}
if(obj==rb3)
{
flag=2;
}
if(obj==rb4)
{
flag=3;
}
}
}
```

SearchMember.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;
public class SearchMember extends JInternalFrame implements ActionListener
```

```
{
JPanel pBook = new JPanel ();
JLabel lbSearch;
JRadioButton rb1,rb2;
JTextField txtSearch;
JButton btnFind, btnCancel;
int flag=0,rows=0;
Statement st;
String mname,mcat,search;
JTable table;
Object data1[][];
Container c;
int mid,bcnt;
public SearchMember (Connection con)
{
super ("Search Members", false, true, false, true);
setSize (325, 250);
lbSearch = new JLabel ("Search Field");
lbSearch.setForeground (Color.black);
lbSearch.setBounds (15, 15, 100, 20);
txtSearch = new JTextField ();
txtSearch.setBounds (120, 15, 175, 25);
btnFind = new JButton ("Find Member");
btnFind.setBounds (25, 175, 125, 25);
btnFind.addActionListener (this);
btnCancel = new JButton ("Cancel");
btnCancel.setBounds (165, 175, 125, 25);
btnCancel.addActionListener (this);
rb1=new JRadioButton("By Id");
rb1.addActionListener(this);
rb1.setBounds (15, 45, 100, 20);
rb2=new JRadioButton("By Name");
rb2.addActionListener(this);
rb2.setBounds (15, 75, 100, 20);
pBook.setLayout (null);
```

```
pBook.add(lbSearch);
pBook.add(txtSearch);
pBook.add(btnFind);
pBook.add(btnCancel);
ButtonGroup bg=new ButtonGroup();
bg.add(rb1);
bg.add(rb2);
pBook.add(rb1);
pBook.add(rb2);
rb1.setSelected(true);
getContentPane().add (pBook, BorderLayout.CENTER);
try
{
st = con.createStatement ();
}
catch (SQLException sqlex) {
JOptionPane.showMessageDialog (null, "A Problem Occurs While Loading Form.");
dispose ();
}
setVisible (true);
}
public void actionPerformed (ActionEvent ae) {
Object obj = ae.getSource();
if (obj == btnFind) {
if (txtSearch.getText().equals ("")) {
JOptionPane.showMessageDialog (this, "Search Field not Provided.");
txtSearch.requestFocus ();
}
else
{
String mname1;
int num,id,catid,bcnt1;
boolean found = false;
ResultSet rs,rs1,rs3;
try {
```

```
String bavl,text,tts;
num=st.executeUpdate("Delete * from MSearch");
if(flag==0)
{
id=Integer.parseInt(txtSearch.getText());
rs=st.executeQuery("Select * from Members where id="+id+"");
rs.next();
bavl=rs.getString("Mname");
catid=rs.getInt(7);
bcnt=rs.getInt(5);
s1=st.executeQuery("Select * from MeCat where Mcat="+catid+"");
rs1.next();
mcat=rs1.getString("CName");
bcnt1=rs1.getInt("Blmt");
rs3=st.executeQuery("Select * from Books where Mid="+id+"");
text="Name: "+bavl+"\n Category: "+mcat+"\n Books Held: "+bcnt+"\n Book Limit: "+bcnt1+"\n";
text+="Books Held:\n";
while(rs3.next())
{
tts=rs3.getString(2);
text+=tts+"\n";
}
JOptionPane.showMessageDialog(this,text);
txtSearch.setText("");
txtSearch.requestFocus();
}
else
{
search=txtSearch.getText();
search=search.toLowerCase();
rs=st.executeQuery("Select * from Members");
while(rs.next())
{
mname=rs.getString(3);
mid=rs.getInt(1);
```

```
bcnt=rs.getInt(5);
catid=rs.getInt(7);
if(flag==1)
{
mname1=mname.toLowerCase();
if(mname1.equals(search)||(mname1.indexOf(search)!=-1))
{
rs1=st.executeQuery("Select * from MeCat where Mcat="+catid+"");
rs1.next();
mcat=rs1.getString("CName");
bcnt1=rs1.getInt("Blmt");
num=st.executeUpdate("insert into MSearch values("+mid+", '"+mname+"' ,"+bcnt+", '"+mcat+"',"+bcnt1+")");
rows++;
found=true;
}
}
}
}
catch(SQLException sqlex) {
if (found == false) {
JOptionPane.showMessageDialog (this, "Record not Found.");
}
}if(flag==1){
try{
data1=new Object[rows][5];
Object[] Colheads={"Member Id","Name","Books Held","Category","Book Limit"};
ResultSet rs2=st.executeQuery("Select * from MSearch");
for(int i1=0;i1<rows;i1++)
{
rs2.next();
for(int j1=0;j1<5;j1++)
{
data1[i1][j1]=rs2.getString(j1+1);
}
}
```

```
table=new JTable(data1,Colheads);
TableDisp td=new TableDisp(table);
txtSearch.setText("");
txtSearch.requestFocus();
}
catch(Exception sqlex) {
if (found == false) {
JOptionPane.showMessageDialog (this, "Some prob Found.");
}
}
}
}
}
if (obj == btnCancel) {
setVisible (false);
dispose();
}
if(obj==rb1)
{
flag=0;
}
if(obj==rb2)
{
flag=1;
}
}
}
```

TableDisp.java

```
import java.awt.*;
import javax.swing.*;
public class TableDisp extends JFrame {
private JPanel pBook = new JPanel ();
private JScrollPane scroller;
```

```
private JTable table;
public TableDisp(JTable j){
super("Table Display");
setSize(500,300);
pBook.setLayout (null);
table=j;
scroller = new JScrollPane (table);
scroller.setBounds (20, 50, 460, 200);
pBook.add(scroller);
getContentPane().add (pBook, BorderLayout.CENTER);
setVisible(true);
}
}
```