**SOUTH EASTERN RAILWAY**

**A REPORT ON**

**DUTY ROSTER USING JAVA**

**BY**

**SOUJANNYA ROY**

**INSTITUTE OF ENGINEERING AND MANAGEMENT**

**KOLKATA**

**ACKNOWLEDGEMENT**

We are highly thankful to **Mr. Kanchan Kumar Kundu**- Sr. Engineer (IT) for this valuable assistance in the completion of our training on “Database Management” in South Eastern Railway.

Next we would like to thank **Mr. Ajay Kumar-**Sr. EDP Manager of South Eastern Railway for guidance, advices, support and allowing us to undergo training in the premises of South Eastern Railway Headquarter.

Lastly, we would also like to acknowledge the authority of South Eastern Railway for providing an industrial environment and facilities required to complete the project and cooperating with us. We worked as a team throughout these days and ultimately ended up with satisfactory results.

Thanking you,

SOUJANNYA ROY

**PREFACE**

This report contains the work done during the winter training done at South Eastern Railway, Garden Reach under the supervision of Mr Kanchan Kumar Kundu. The report will give an overview of the tasks completed during the period of training with the technical details. The tasks done then will be discussed and analysed.

Report will also elaborate on future works that can be persuaded as an advancement of current work.

We have tried to keep the report simple yet technically correct. We hope we have succeeded in our attempt.

**Soujannya Roy**

**INDEX**

|  |  |
| --- | --- |
| **Sl.No.** | **CONTENT** |
| 1. | INTRODUCTION |
| 2. | PURPOSE OF THE PROJECT |
| 3. | TECHNOLOGY USED |
| 4. | FLOWCHART OF THE PROJECT |
| 5. | SOURCECODES AND OUTPUTS |
| 6. | CONCLUSION |
| 7. | BIBLIOGRAPHY |

**INTRODUCTION**

Duty rosters are schedules that assign the shifts and tasks to the employees. It can be used to assign alternative work to a group of people belonging to the same team. It makes sure that only necessary people remain at one shift to manage others at different time slots.

A good roster plan (or employee shift schedule) ensures that each shift has enough employees to keep things running smoothly and efficiently. A good employee schedule also prevents anyone from getting into an accident at work because of exhaustion, which means your roster plan can also be reinforced to keep your employees safe and healthy. A great employee shift schedule also prevents disorganisation and wastefulness.

Another good reason to create a good roster plan is to determine whether you have got underperforming employees, and whether such poor performances have anything to do at all with their skills and training or that it simply is a matter of poor scheduling. Some employees might be better suited for the night shift while others for day or mid shift. The ideal roster design can enable you to spot any issues that generate problems for certain shifts and, consequently, for the smooth functioning of the office.

The implementation of this roster is done by using Java for assigning employees of the I.T console South Eastern Railway in shifts for the smooth working of the sub-department.

**PURPOSE OF THE PROJECT**

* This project can be used to make duty roster to assign shifts to the employees in South Eastern Railway, Head Quarter, IT Centre.
* The duty roster is displayed in form of a table which is editable and printable.
* We can also export it to an Excel sheet.
* All the information in the table can be updated when it is required.
* The code is flexible ,it allows the administrator to make changes easily when there are changes in the form of joining or retirement of employees ,and also takes into account the preferred shift of each employee.
* We can make duty roster for any number of months ahead.
* Once a new roster is made for a particular month ,is will be saved and can be edited anytime without entering all the data from the start.

**TECHNOLOGY USED**

**Language: java**

**GUI Toolkit for JTable: SWING**

**JAVA:** Java programming language was originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform (Java 1.0 [J2SE]).

The latest release of the Java Standard Edition is Java SE 8. With the advancement of Java and its widespread popularity, multiple configurations were built to suit various types of platforms. For example: J2EE for Enterprise Applications, J2ME for Mobile Applications.

Java is guaranteed to be **Write Once, Run Anywhere.**

Java is −

* **Object Oriented** − In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
* **Platform Independent** − Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.
* **Simple** − Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.
* **Secure** − With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
* **Architecture-neutral** − Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.
* **Portable** − Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.
* **Robust** − Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.
* **Multithreaded** − With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.
* **Interpreted** − Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.
* **High Performance** − With the use of Just-In-Time compilers, Java enables high performance.
* **Distributed** − Java is designed for the distributed environment of the internet.
* **Dynamic** − Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

We have used JDK jdk1.8.0\_131 for this project.

The roster will be displayed in the form of a JTable which is a part of Swing GUI Toolkit in Java.

**Swing** is a [GUI](https://en.wikipedia.org/wiki/Graphical_user_interface) [widget toolkit](https://en.wikipedia.org/wiki/Widget_toolkit) for [Java](https://en.wikipedia.org/wiki/Java_(programming_language)). It is part of [Oracle](https://en.wikipedia.org/wiki/Oracle_Corporation)'s [Java Foundation Classes](https://en.wikipedia.org/wiki/Java_Foundation_Classes) (JFC) – an [API](https://en.wikipedia.org/wiki/Application_programming_interface) for providing a [graphical user interface](https://en.wikipedia.org/wiki/Graphical_user_interface) (GUI) for Java programs.

Swing provides a [look and feel](https://en.wikipedia.org/wiki/Look_and_feel) that emulates the look and feel of several platforms, and also supports a [pluggable look and feel](https://en.wikipedia.org/wiki/Pluggable_look_and_feel) that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

Unlike AWT components, Swing components are not implemented by platform-specific code. Instead, they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element.

The JTable class is a part of Java Swing Package and is generally used to display or edit two-dimensional data that is having both rows and columns. It is similar to a spreadsheet. This arranges data in a tabular form.

*Constructors in JTable:*

1. JTable(): A table is created with empty cells.
2. JTable(int rows, int cols): Creates a table of size rows \* cols.
3. JTable(Object[][] data, Object []Column): A table is created with the specified name where []Column defines the column names.

*Functions in JTable:*

1. addColumn(TableColumn []column) : adds a column at the end of the JTable.
2. clearSelection() : Selects all the selected rows and columns.
3. editCellAt(int row, int col) : edits the intersecting cell of the column number col and row number row programmatically, if the given indices are valid and the corresponding cell is editable.
4. setValueAt(Object value, int row, int col) : Sets the cell value as ‘value’ for the position row, col in the JTable.

**FLOWCHART OF THE PROJECT**

Yes

No

Yes

open file

Check rest allocation

Enter allocation

Enter data

d

data

Edit Table

No

Checking for rest allotment of employees

Leave Schedule

Enter Gazetted holidays

Input month and year and names of employees and shifts

Previous Roster Available?

yes

Yes

Edit Gazetted holidays

Enter gazetted holidays

No

Display table and save file

**SOURCE CODE AND OUTPUT:**

***CODE:***

**import** java.text.DateFormat;

**import** java.text.MessageFormat;

**import** java.text.SimpleDateFormat;

**import** java.util.\*;

**import** java.util.List;

**import** javax.swing.\*;

**import** javax.swing.table.JTableHeader;

**import** javax.swing.table.TableColumn;

**import** javax.swing.table.TableModel;

**import** java.awt.\*;

**import** java.io.\*;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.print.PrinterException;

**public** **class** calender **extends** JFrame

{

**public** calender(String str\_date,**int** res,**int**[]day,String[]day\_of\_week,String[] morning\_shift,String[] day\_shift,String[] evening\_shift,String []rest,String[]leave)

{

**super**("DUTY ROSTER FOR "+str\_date);

String sday[]=**new** String[res];

**for**(**int** i=0;i<res;i++)

sday[i]= Integer.*toString*(day[i]);

setDefaultCloseOperation(***EXIT\_ON\_CLOSE***);

Object[][] data=**new** Object[res][7];

**for**(**int** i=0;i<res;i++)

{

data[i][0]=sday[i];

data[i][1]=day\_of\_week[i];

data[i][2]=morning\_shift[i];

data[i][3]=day\_shift[i];

data[i][4]=evening\_shift[i];

data[i][5]=rest[i];

data[i][6]=leave[i];

}

String[] ch= {"DATE","DAY","MORNING SHIFT","DAY SHIFT","EVENING SHIFT","REST","LEAVE"};

JTable table=**new** JTable(data,ch);

table.setPreferredScrollableViewportSize(getMinimumSize());

//table.setPreferredSize(new Dimension(100,450));

TableColumn col0=table.getColumnModel().getColumn(0);

TableColumn col1=table.getColumnModel().getColumn(1);

TableColumn col2=table.getColumnModel().getColumn(2);

TableColumn col3=table.getColumnModel().getColumn(3);

TableColumn col4=table.getColumnModel().getColumn(4);

TableColumn col5=table.getColumnModel().getColumn(5);

TableColumn col6=table.getColumnModel().getColumn(6);

col0.setPreferredWidth(40);

col1.setPreferredWidth(70);

col2.setPreferredWidth(200);

col3.setPreferredWidth(100);

col4.setPreferredWidth(200);

col5.setPreferredWidth(100);

col6.setPreferredWidth(100);

JButton button;

JButton button1;

JScrollPane js=**new** JScrollPane(table);

getContentPane().add(js);

js.setBounds(180,40,1000,2000);

button=**new** JButton("PRINT");

button.setBounds(570,580,250,30);

button1=**new** JButton("EXPORT TO EXCEL");

button1.setBounds(570,630,250,30);

getContentPane().add(js,BorderLayout.***CENTER***);

JTableHeader ht=table.getTableHeader();

ht.setBackground(Color.*decode*("#b6ddba"));

ht.setFont(**new** Font("Tahome",Font.***BOLD***,14));

button.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e)

{

MessageFormat header=**new** MessageFormat("DUTY ROSTER FOR "+str\_date);

MessageFormat footer=**new** MessageFormat("SR.ENGINEER(IT)/IT CENTRE(console operation)");

**try** {

table.print(JTable.PrintMode.***FIT\_WIDTH***, header, footer);

} **catch** (PrinterException e1) {

// **TODO** Auto-generated catch block

JOptionPane.*showInternalMessageDialog*(**null**, "unable to print");

}

}

});

button1.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e)

{ String f=str\_date.replace("/",",");

String a=

"C:\\Users\\User\\Desktop\\"+f+".xls\\";

File file=**new** File(a);

**try**{

TableModel model = table.getModel();

FileWriter excel = **new** FileWriter(file);

**for**(**int** i = 0; i < model.getColumnCount(); i++){

excel.write(model.getColumnName(i) + "\t");

}

excel.write("\n");

**for**(**int** i=0; i< model.getRowCount(); i++) {

**for**(**int** j=0; j < model.getColumnCount(); j++) {

excel.write(model.getValueAt(i,j).toString()+"\t");

}

excel.write("\n");

}

excel.close();

}**catch**(IOException e1){ System.***out***.println(e1); }}

});

add(button);

add(button1);

add(js);

setLayout(**null**);

setSize(1500,1500);

getContentPane().setBackground(Color.*decode*("#bdb76b"));

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

setLocationRelativeTo(**null**);

setVisible(**true**);

}

**private** **static** Scanner *sc*;

**public** **static** **int** *k*=0;

**public** **static** File *temp*;

**public** **static** **void** main(String ar[]) **throws** FileNotFoundException {

**int** c=0,p=0,z=0;

*sc* = **new** Scanner(System.***in***);

Calendar now = Calendar.*getInstance*();

**int** y1=now.get(Calendar.***YEAR***);

// month start from 0 to 11

**int** m1=now.get(Calendar.***MONTH***) + 1;

System.***out***.println("enter month number and year for required roster as mmyyyy");

String u=*sc*.nextLine();

String u1=u.substring(0,2);

String u2=u.substring(2);

**int** m2=Integer.*valueOf*(u1);

**int** y2=Integer.*valueOf*(u2);

**if**(m2<1 || m2>12)

{

System.***out***.println("WRONG INPUT!!!! Month should be between 1 and 12\nExecute again");

System.*exit*(0);

}

**int** md,yd;

md=m2-m1;

yd=y2-y1;

md=md+yd\*12;

now.add(Calendar.***MONTH***, md);

Date date = now.getTime();

DateFormat dateFormat = **new** SimpleDateFormat("MM/yyyy");

String strDate = dateFormat.format(date);

System.***out***.println("Calender for:"+strDate);

now.set(Calendar.***DATE***,1);

String[] strDays = **new** String[] { "Sun", "Mon", "Tue", "Wed", "Thu",

"Fri", "Sat" };

System.***out***.print("Enter number of employees:");

**int** l=*sc*.nextInt();

String[] emp\_name = **new** String[l];

// Day\_OF\_WEEK starts from 1 while array index starts from 0

**int** max\_rest=0;

**int** res = now.getActualMaximum(Calendar.***DATE***);

**int** day[]=**new** **int**[res];

String day\_shift[]=**new** String[res];

String morning\_shift[]=**new** String[res];

String evening\_shift[]=**new** String[res];

String rest[]=**new** String[res];

String leave[]=**new** String[res];

String[] day\_of\_week=**new** String[res];

String[] shifts=**new** String[res] ;

**int**[] holiday\_list=**new** **int**[res];

c=(now.get(Calendar.***DAY\_OF\_WEEK***)-1);

z=c;

p=c;

String f1=strDate.replace("/",",");

String filename1 = "roster\_for\_"+f1+".file";

*temp*=**new** File(filename1);

**if**(*temp*.exists()==**false**)

{

*sc*.nextLine();

System.***out***.println("Assign shifts to employees\nEnter names of employees in morning shift:(Press 0 to exit)");

**int** xy,cm=0;

**for**(**int** i=0;i<emp\_name.length;i++) {

String s=*sc*.nextLine();

**if**(s.equals("0")==**false**) {

emp\_name[cm++]=s;

}

**else** **break**;

}

//System.out.println(cm);

xy=l-cm;

System.***out***.println("Enter names of employees in evening shift:");

**int** b=cm;

**for**(**int** i=cm;i<emp\_name.length;i++) {

String s=*sc*.nextLine();

emp\_name[b++]=s;

}

String cm1=Integer.*toString*(cm);

String f=strDate.replace("/",",");

//String filename="roster\_for\_"+f+".file";

String f2="emp"+f+".txt";

BufferedWriter outputWriter = **null**;

**try** {

outputWriter = **new** BufferedWriter(**new** FileWriter(f2));

outputWriter.write(cm1);

**try** {

outputWriter.newLine();

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

} **catch** (IOException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

**for** (**int** i = 0; i <l; i++)

{

// Maybe:

**try** {

outputWriter.write(emp\_name[i]+"");

} **catch** (IOException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

**try** {

outputWriter.newLine();

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

**try** {

outputWriter.flush();

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**try** {

outputWriter.close();

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**for**(**int** i=0;i<res;i++)

{

day[i]=i+1;

day\_of\_week[i]=strDays[c];

**if**((c==6 && i<=13 && i>=7)||c==0)

{

day\_shift[i]="Rest to all";

max\_rest++;

}

**else**

day\_shift[i]=" ";

c++;

**if**(c>6)

c=0;

}

*hol*(cm,holiday\_list,day,day\_shift,morning\_shift,evening\_shift,res,emp\_name,day\_of\_week);

**for**(**int** i=0;i<res;i++)

{

morning\_shift[i]="";

evening\_shift[i]="";

}

**for**(**int** i=0;i<res;i++)

{

**if**(day\_shift[i]!="Rest to all")

{

**for**(**int** j=0;j<cm;j++) {

morning\_shift[i]=emp\_name[j]+" "+morning\_shift[i];

//System.out.println(morning\_shift[i]);

}

**for**(**int** j=cm;j<l;j++) {

evening\_shift[i]=emp\_name[j]+" "+evening\_shift[i];}

}

**else**

{

morning\_shift[i]="\t";

evening\_shift[i]="\t";

}

}

**for**(**int** i=0;i<res;i++)

{

**if**(z==6 && i<=13 && i>=7)

{

day\_shift[i]=rest[i]=" ";

}

z++;

**if**(z>6)z=0;

}

**for**(**int** i=0;i<res;i++)

{

**if**(leave[i]==**null**)leave[i]="";

**if**(rest[i]==**null**)rest[i]=" ";

}

calender c1=**new** calender(strDate,res,day,day\_of\_week,morning\_shift,day\_shift,evening\_shift,rest,leave);

c1.setVisible(**true**);

System.***out***.println("DO YOU WANT TO EDIT THE TABLE:y/n");

**char** v=*sc*.next().charAt(0);

**switch**(v)

{

**case** 'Y':

**case** 'y':System.***out***.println("EDIT TABLE");

*edit*(cm,holiday\_list,max\_rest,shifts,strDate, c1,emp\_name,res,day,day\_of\_week,morning\_shift,day\_shift,evening\_shift,rest,leave);

**break**;

**case** 'N':

**case** 'n':System.***out***.println("table not edited ");

**break**;

**default**: System.***out***.println("WRONG INPUT!!!! \tExecute again");

System.*exit*(0);

}

*save\_table*(strDate,res,day\_of\_week,day,morning\_shift,day\_shift,evening\_shift,rest,leave) ;

}

//modify

**else** {

*sc*.nextLine();

String f=strDate.replace("/",",");

String filename="roster\_for\_"+f+".file";

String f2="emp"+f+".txt";

JTable table;

//File f22=new File(f2);

**try** {

ObjectInputStream ois = **new** ObjectInputStream(**new** FileInputStream(filename));

//ObjectInputStream ois1 = new ObjectInputStream(new FileInputStream(f2));

table = (JTable) ois.readObject();

}

**catch**(Exception e) {

System.***out***.println("Problem reading back table from file: " + filename);

**return**;

}

String[] sday=**new** String[res];String[][]data=**new** String[res][7];

**for**(**int** i=0;i<res;i++)

{

**for**(**int** j=0;j<7;j++)

{

data[i][j]=(String) table.getValueAt(i, j);

}

}

**for**(**int** i=0;i<res;i++)

{

**for**(**int** j=0;j<7;j++)

{

sday[i]=data[i][0];

day\_of\_week[i]=data[i][1];

morning\_shift[i]=data[i][2];

day\_shift[i]=data[i][3];

evening\_shift[i]=data[i][4];

rest[i]=data[i][5];

leave[i]=data[i][6];

}

}

**for**(**int** i=0;i<res;i++)

day[i]= Integer.*valueOf*(sday[i]);

calender c1=**new** calender(strDate,res,day,day\_of\_week,morning\_shift,day\_shift,evening\_shift,rest,leave);

c1.setVisible(**true**);

String token1 = "";

Scanner inFile1 = **new** Scanner(**new** File(f2));

List<String> temps = **new** ArrayList<String>();

**while** (inFile1.hasNext()) {

token1 = inFile1.next();

temps.add(token1);

}

inFile1.close();

String cm1=temps.get(0);

**int** cm = Integer.*valueOf*(cm1);

temps.remove(cm1);

emp\_name = temps.toArray(**new** String[1]);

System.***out***.println("\nDUTY ROSTER EXISTS FOR SELECTED MONTH");

System.***out***.println("Do you want to edit gazzeted holidays:y/n");

String h=*sc*.nextLine();

**switch**(h)

{

**case** "Y":

**case** "y":*hol*(cm,holiday\_list,day,day\_shift,morning\_shift,evening\_shift,res,emp\_name,day\_of\_week);

**case** "N":

**case** "n":*edit*(cm,holiday\_list,max\_rest,shifts,strDate, c1,emp\_name,res,day,day\_of\_week,morning\_shift,day\_shift,evening\_shift,rest,leave);

*save\_table*(strDate,res,day\_of\_week,day,morning\_shift,day\_shift,evening\_shift,rest,leave) ;

**break**;

**default**:System.***out***.println("WRONG INPUT!!!!:ENTER y/n");

System.*exit*(0);

**break**;

}

}

}

**public** **static** **void** rst(String[]day\_of\_week,**int**[] sat\_worked,**int** max\_rest,**int** res,String[] emp\_name,String[] shifts,**int**[]day,String[] morning\_shift,String[] day\_shift,String[] evening\_shift,String []rest,String[]leave)

{

**int** res\_des=max\_rest+*k*;

**int**[] res\_taken=**new** **int**[emp\_name.length];

**int** yy=emp\_name.length;

**int**[] leave\_of\_each= **new** **int**[yy];

**for**(**int** i=0;i<leave\_of\_each.length;i++)

{

leave\_of\_each[i]=0;

}

**for**(**int** i=0;i<leave.length;i++)

{

**for**(**int** j=0;j<emp\_name.length;j++)

{

**if**(leave[i].equals(emp\_name[j]))

leave\_of\_each[j]++;

}

}

**for**(**int** i=0;i<leave\_of\_each.length;i++)

{

res\_taken[i]=leave\_of\_each[i]+max\_rest+*k*-sat\_worked[i];

}

**int** gg=0;

**for**(**int** z=0;z<res;z++) {

**if**(day\_of\_week[z].equals("Sat")&& z>=7 && z<=13){gg=z+1;}}

**for**(**int** i=0;i<leave\_of\_each.length;i++)

{

**if**(res\_taken[i]<res\_des)

{

System.***out***.println(emp\_name[i]+" has pending rest,assign rest on:");

**int** dd=*sc*.nextInt();

**if**(dd>gg) {

rest[dd-1]=emp\_name[i];

}

**else** {System.***out***.println("WRONG INPUT!!!!REST CANNOT BE ASSIGNED\n IF EMPLOYEE HAS NOT ALREADY WORKED ON 2ND SATURDAY\nEXECUTE AGAIN(Assign rest after 2nd Saturday)");

System.*exit*(0);}

}

}

**for**(**int** i=0;i<res;i++)

{

**if** (morning\_shift[i].contains(rest[i])) {

morning\_shift[i] = morning\_shift[i].replaceAll(rest[i], " ").trim();

}

**if** (evening\_shift[i].contains(rest[i])) {

evening\_shift[i] = evening\_shift[i].replaceAll(rest[i], " ").trim();

}

}

}//end of res

**public** **static** **void** edit(**int** cm,**int**[]holiday\_list,**int** max\_rest,String[]shifts,String strDate,calender c1,String[] emp\_name,**int** res,**int**[]day,String[]day\_of\_week,String[] morning\_shift,String[] day\_shift,String[] evening\_shift,String []rest,String[]leave)

{calender c2;

System.***out***.println("ASSIGN SHIFTS TO EMPLOYEES(press 0 to exit)");

**int** x=1;

**int**[] sat\_w= **new** **int**[emp\_name.length];

**for**(**int** i=0;i<sat\_w.length;i++)

{sat\_w[i]=0;}

calender c3 = **null**;**int** w=0;

**while**(x!=0)

{

System.***out***.print("enter day for which shift needs to be edited: ");

x=*sc*.nextInt();

**if**(x==0)**continue**;

*sc*.nextLine();

**for**(**int** z=0;z<res;z++) {

**if**(day\_of\_week[x-1].equals("Sat")&& z>=7 && z<=13){

day\_shift[x-1]=" ";

**for**(**int** i=0;i<res;i++)

{

**if**(rest[i]!=" ")

{

**for**(**int** r=0;r<emp\_name.length;r++) {

**if**(rest[i].equals(emp\_name[r])&&r<cm)

{morning\_shift[i]=rest[i]+" "+morning\_shift[i];rest[i]=" ";}

**else** **if**(rest[i].equals(emp\_name[r])&&r>=cm)

{evening\_shift[i]=rest[i]+" "+evening\_shift[i];rest[i]=" ";}

}

}}}}

c3=**new** calender(strDate,res,day,day\_of\_week,morning\_shift,day\_shift,evening\_shift,rest,leave);

c1.setVisible(**false**);

c3.setVisible(**true**);

w=1;

**for**(**int** i=0;i<res;i++)

{

**if**(day[i]==x)

{

morning\_shift[i]=" ";

day\_shift[i]=" ";

evening\_shift[i]=" ";

leave[i]=" ";

System.***out***.println("Enter allocation for the employees");

System.***out***.println("m-Morning\td-Day\te-Evening\tl-Leave\t\tPress ENTER for no allocation");

**for**(**int** j=0;j<emp\_name.length;j++)

{

System.***out***.println("Employee name:"+emp\_name[j]);

System.***out***.print("Assign shift:");

shifts[j]=*sc*.nextLine();

**if**(shifts[j].equals("m"))

{

morning\_shift[i]=emp\_name[j]+" "+morning\_shift[i];

}

**if**(shifts[j].equals("d"))

{

sat\_w[j]++;

day\_shift[i]=emp\_name[j]+" "+day\_shift[i];

}

**if**(shifts[j].equals("e"))

{

evening\_shift[i]=emp\_name[j]+" "+evening\_shift[i];

}

**if**(shifts[j].equals("l"))

{

leave[i]=emp\_name[j]+" "+leave[i];

}

}

}

}

}

*rst*(day\_of\_week,sat\_w,max\_rest,res,emp\_name,shifts,day,morning\_shift,day\_shift,evening\_shift,rest,leave);

c2=**new** calender(strDate,res,day,day\_of\_week,morning\_shift,day\_shift,evening\_shift,rest,leave);

c2.setVisible(**true**);

**if**(w==1)

c3.setVisible(**false**);

c1.setVisible(**false**);

}

**public** **static** **void** save\_table(String strDate,**int** res,String[] day\_of\_week,**int**[]day,String[]morning\_shift,String []day\_shift,String[] evening\_shift,String[]rest,String[]leave)

{

String sday[]=**new** String[res];

**for**(**int** i=0;i<res;i++)

sday[i]= Integer.*toString*(day[i]);

Object[][] data=**new** Object[res][7];

**for**(**int** i=0;i<res;i++)

{

data[i][0]=sday[i];

data[i][1]=day\_of\_week[i];

data[i][2]=morning\_shift[i];

data[i][3]=day\_shift[i];

data[i][4]=evening\_shift[i];

data[i][5]=rest[i];

data[i][6]=leave[i];

}

String[] ch= {"DATE","DAY","MORNING SHIFT","DAY SHIFT","EVENING SHIFT","REST","LEAVE"};

JTable table=**new** JTable(data,ch);

String f=strDate.replace("/",",");

String filename = "roster\_for\_"+f+".file";

**try** {

ObjectOutputStream oos = **new** ObjectOutputStream(**new** FileOutputStream(filename));

oos.writeObject(table);

oos.close();

}

**catch**(IOException e) {

System.***out***.println("Problem creating table file: " + e);

**return**;

}

System.***out***.println("JTable correctly saved to file " + filename);

}

**public** **static** **void** hol(**int** cm,**int**[] holiday\_list,**int**[]day,String[]day\_shift,String[]morning\_shift,String[]evening\_shift,**int** res,String[] emp\_name,String[]day\_of\_week) {

**if**(*temp*.exists()) {

**int** b=0;

**for**(**int** i=0;i<res;i++)

{

**if**(day\_shift[i].equals("Rest to all")) {

**if**( day\_of\_week[i].equals("Sun")==**false**) {

holiday\_list[b++]=day[i];

}

}

*k*=b;

}

}

**for**(**int** i=0;i<res;i++)

{

**for**(**int** j=0;j<*k*;j++)

{

**if**(day[i]==holiday\_list[j]) {

day\_shift[i]="\t";

**for**(**int** m=0;m<cm;m++) {

morning\_shift[i]=emp\_name[m]+" "+morning\_shift[i];

}

**for**(**int** m=cm;m<emp\_name.length;m++) {

evening\_shift[i]=emp\_name[j]+" "+evening\_shift[i];}

}

}

}

**for**(**int** i=0;i<*k*;i++)

{

holiday\_list[i]=0;

}

System.***out***.println("\nEnter the date of the gazetted holidays in the month: (PRESS 0 TO EXIT)");

**while**(*k*<res)

{ **int** a=*sc*.nextInt();

**if**(a!=0)

{holiday\_list[*k*++]=a;

}

**else**

**break**;

}

**for**(**int** i=0;i<res;i++)

{

**for**(**int** j=0;j<*k*;j++)

{

**if**(day[i]==holiday\_list[j]) {

day\_shift[i]="Rest to all";

morning\_shift[i]="\t";

evening\_shift[i]="\t";}

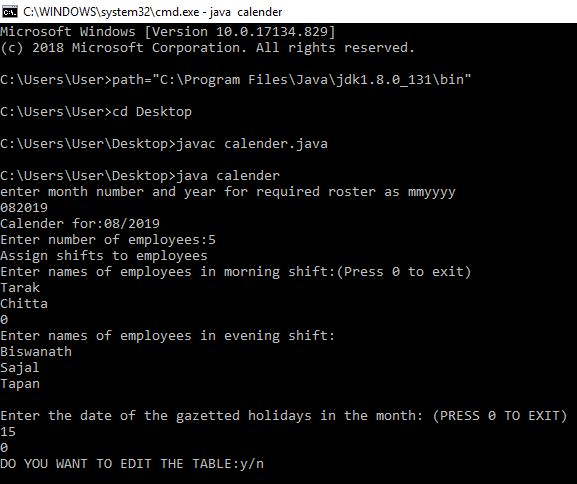
}

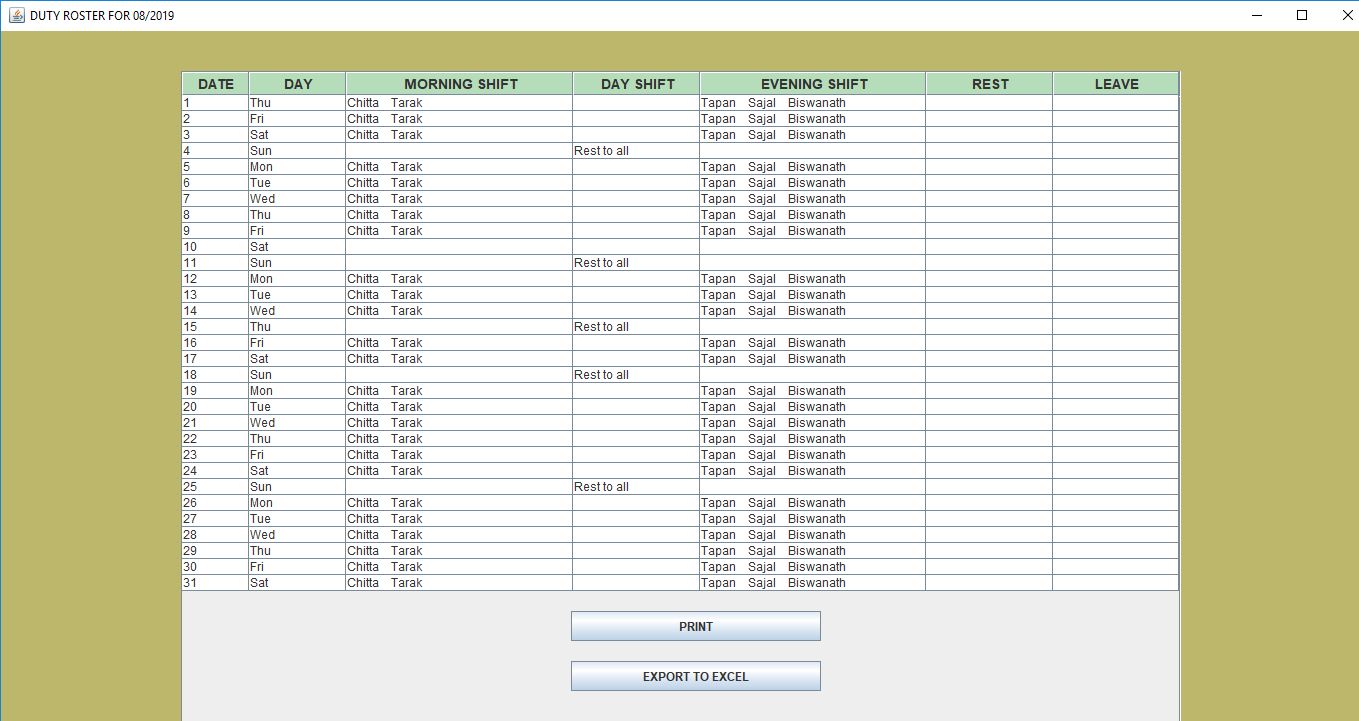
}

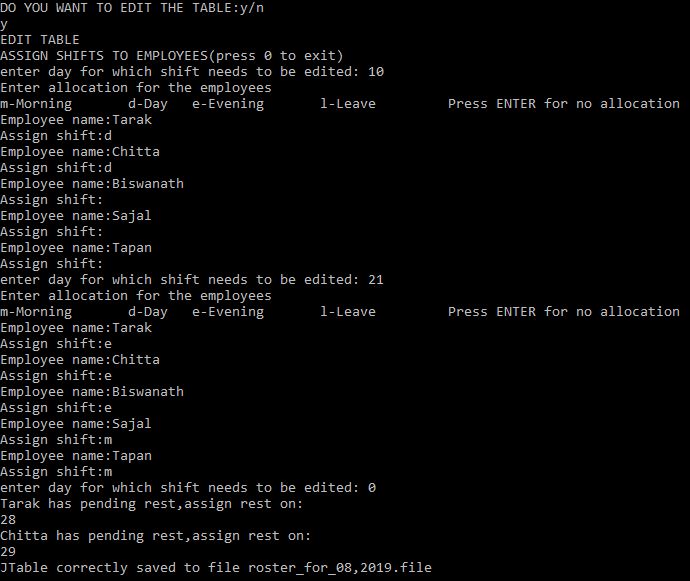
}

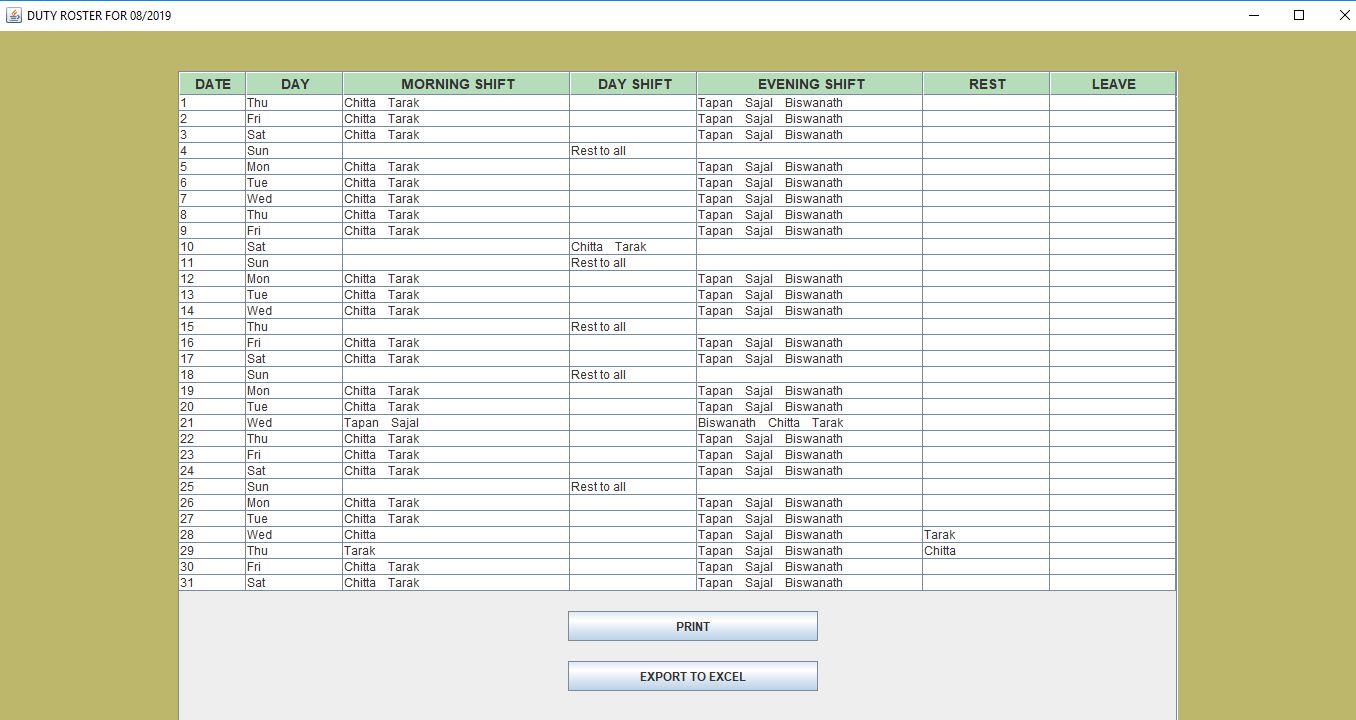
}

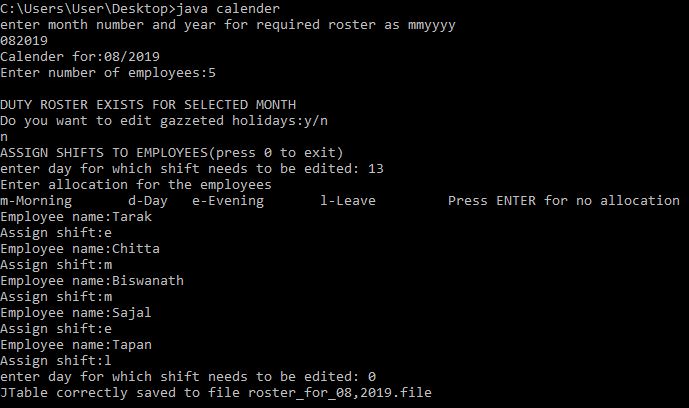
***OUTPUT:***

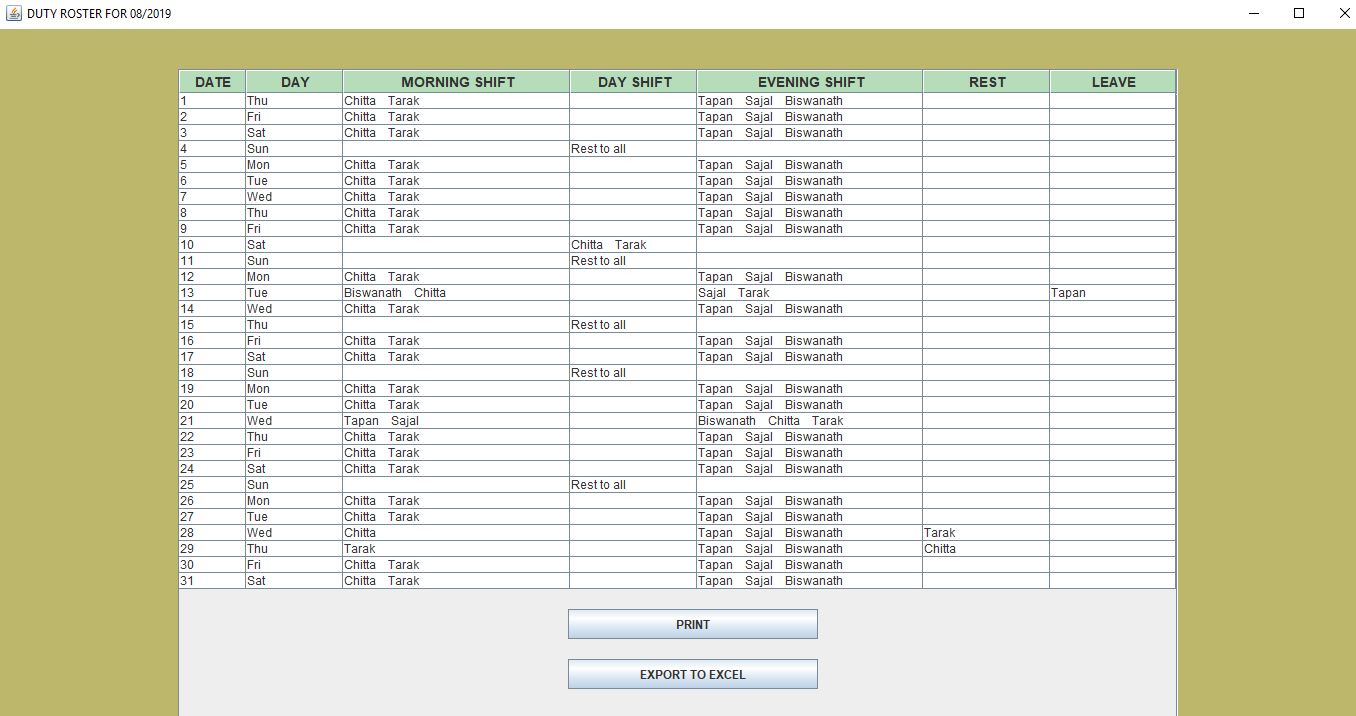












**Conclusion**

This project is primarily aimed to create a duty roster for the shift allocation of the employees of the I.T Console, South Eastern Railway Headquarters. The implementation of this is done by using Java a powerful dynamic programming language. The program is made as dynamic as possible, to simplify the usage and making it more reliable and efficient.

When staff shifts meet the needs of operations, the resources are maximised and significantly improve day-to-day operations, both of which will always translate into better resource sharing and productivity.

The age-old conventional method of creating a roster table by hand or printed medium is tenacious, and time consuming, using this programming solution will no doubt save time and is free from errors humans make.

The solution is expected to be used by the Senior Engineer IT Console for creating the employee shift allocation, duty roster for a particular month and later edit it according to his requirments.

In future, this project can be designed to provide a full Graphical User Interface which will be more user friendly .This can also be made into an app so that the admin can make rosters from his mobile phone.

**BIBLIOGRAPHY**

We have collected the information from the following sources:

* [www.youtube.com](http://www.youtube.com)
* <https://www.geeksforgeeks.org/java-swing-jtable/>
* <https://en.wikipedia.org/wiki/Swing_(Java)>
* www.tutorialspoint.com/java
* [www.dreamincode.net/forums](http://www.dreamincode.net/forums)
* [www.w3schools.com](http://www.w3schools.com)
* [www.javatpoint.com](http://www.javatpoint.com)
* [www.stackoverflow.com](http://www.stackoverflow.com)

**Declaration**

We hereby declare that training entitled “Duty Roster Using Java” is an authentic record of our own as a record of three weeks training during period from 18.06.2019 to 09.07.2019 under guidance of Mr. Kanchan Kumar Kundu – Sr. Engineer, Mr. Ajay Kumar- Sr. EDP manager.

By completing the project, we succeeded to fulfil the requirements as asked by the authority. The project is completed and successfully running.

SUBMITED BY: -

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | UNIVERSITY ROLL | DEPARTMENT | INSTITUTION |
| SOUJANNYA ROY | 10400116090 | CSE | INSTITUTE OF ENGINEERING AND MANAGEMENT |