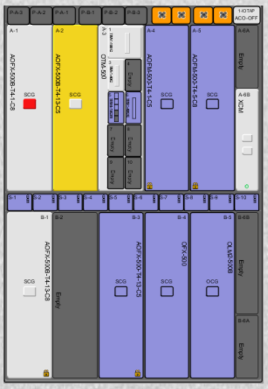
Dynamic Sorting of Chassis in the Chassis View of DNA

By Rahul S Agasthya



CONTENTS

ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and deep regards to my guide Sravanthi Konduru for her exemplary guidance, monitoring and constant encouragement throughout the course of this project. The help and guidance given by her time to time shall carry me a long way in the journey of life on which I am about to embark.

I also take this opportunity to express a deep sense of gratitude to Yashpal Kumar and Bharat Koratikere Bhaskaraiah, for their cordial support, valuable information and guidance, which helped me in completing this task through various stages.

I am obliged to staff members of Infinera India Private Limited, for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

Lastly, I thank almighty, my parents, sister and friends for their constant encouragement without which this assignment would not be possible.

- Rahul S. Agasthya

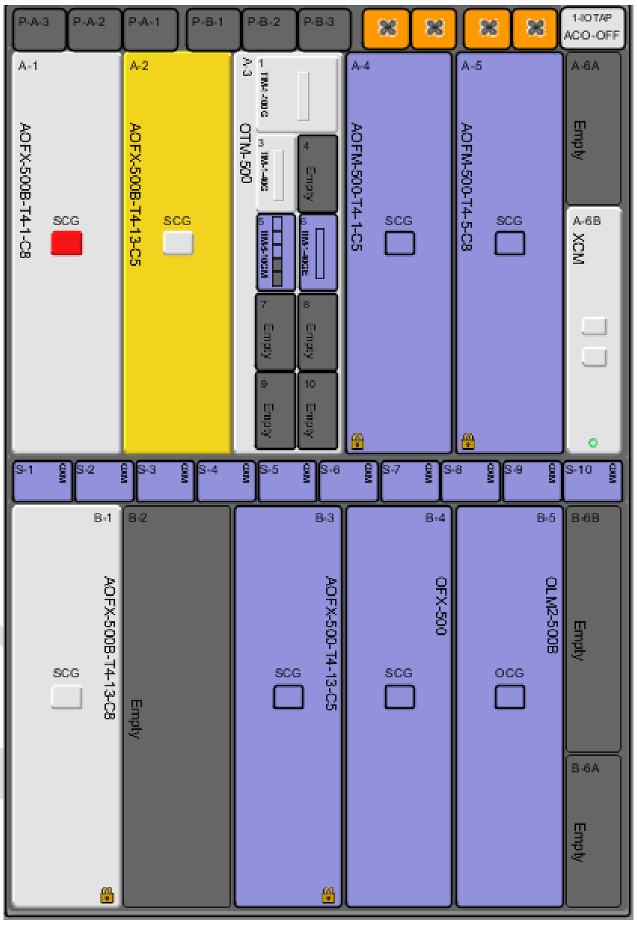
INTRODUCTION

➢ THE DATA TRANSPORT NETWORK

SCALABLE
• 500 Gb/s super-channel
• 5 Tb/s with future upgrade to 100 Tb/s
SIMPLE
• Intelligent GMPLS automation
• Converged DWDM/OTN; MPLS future
EFFICIENT
• 50% less power
• 33% less space
infinera

- Designed to address the ever-evolving needs of the core network, the DTN-X leverages the latest generation of PIC technology to deliver the industry's fastest, simplest and most efficient Packet Optical Transport Platform.
- Through a unique combination of innovative architecture and cutting-edge technology, the DTN-X brings revenue to network operators faster, simpler and more efficiently than any of the leading competitors. Sized to fit your application needs, the DTN-X is built to scale in multiple dimensions without sacrifice.
- DTN-X integrates the world's highest density 500 Gb/s PICs, upgradeable in the future to 1Tb/s, and offers 5 Tb/s of switching capacity today with up to 100Tb/s of capacity in the future that can be used for any mix of optical transport, OTN (ITU G.709) and in the future MPLS switching.
- In addition, the DTN-X automates many traditional network engineering steps, so operators spend less time engineering the network and more time delivering services and generating revenue.
- Further, with built-in GMPLS-based intelligence, the network is always aware and able to tap into available resources, optimal routing paths and most importantly, mesh-based network protection that protects customer traffic, even in the event of catastrophic failures as fast as <50ms.
- Finally, sample operator network modeling has shown that DTN-X can save as much 50% in power and 33% in space over competing systems.

➢DIGITAL NETWORK ADMINISRTRATOR



- The Infinera DNA combines node-level and network-wide management in one integrated software system, making it easy to simplify network operations, speed service provisioning, and rapidly isolate problems anywhere they occur.

- The Infinera Digital Network Administrator (DNA) is a carrier-class management system that offers extensive fault, configuration, performance, and security management capabilities across multiple Infinera network elements and subnetworks.

- The Infinera DNA works with Infinera's intelligent Generalized Multi-Protocol Label Switching auto-discovery and auto-provisioning software to enable rapid point-and-click provisioning of managed transparent wave services, ranging from 155/622Mb/s, 1Gb/s, 2.5Gb/s, 10Gb/s and beyond, using a feature-rich GUI.

- Based on multi-tiered server architecture, the Infinera DNA can scale to manage thousands of network elements and hundreds of users, and can support multiple administrative partitions for customized management domains.

- Context-sensitive navigation, integrated online help, powerful debugging tools, easy-to-use service templates, and a wide range of inventory and performance reports are all available to further simplify operations and administrative tasks.

THE ASSIGNMENT

In the Network Element's chassis view, when the chassis were added to the rack, the chassis would not get sorted in order of their Rack Unit Location.
They would get jumbled in a haphazard manner.

My assignment was to ensure that the chassis get sorted every time a new Chassis is introduced into the rack of the Network Element.

My solution to this issue was considering the chassis as an object of a class. Using the Rack Unit Location, the objects are fed into a List. Sort the list and create a block to represent the chassis in the rack according to the sorted list.

Objective:

Upon running the Java swing code, a window gets created having three empty racks by default. These empty racks are colour coded grey to show that they are empty.

At the bottom of the window there is a button called as "Add Chassis". Upon click this button, a new Dialog box pops up. The user will have to enter the Rack Number and the Rack Unit Location.

Add the bottom of this dialog box, is present a button called as "Create Chassis". Upon clicking this button, the Chassis created will be placed in the rack specified.

Note:
Every time a chassis is created, it is placed in the rack as per its Rack Unit Location. The rack is always sorted.

The number of racks is limited, but the number of chassis per rack is not. The newly added chassis will follow a grid layout when displayed on the parent window.

PROJECT REQUIREMENTS

➢ PROGRAMMING LANGUAGE

Java

➢ EDITOR USED

IntelliJ IDEA
IntelliJ IDEA is a Java IDE by JetBrains, available as an Apache 2 Licensed community edition and a commercial edition.
In a report by Infoworld in 2010, IntelliJ got the highest test center score out of the 4 Top Java Programming Tools :

- Eclipse
- IntelliJ IDEA
- NetBeans
- Oracle JDeveloper.

➢ JAVA DEVELOPMENT KIT

The **Java Development Kit** (**JDK**) is an implementation of either one of the Java SE, Java EE or Java ME platforms released by Oracle Corporation in the form of a binary product aimed at Java developers on Solaris, Linux, Mac OS X or Windows. Since the introduction of the Java platform, it has been by far the most widely used Software Development Kit (SDK). On 17 November 2006, Sun announced that it would be released under the GNU General Public License(GPL), thus making it free software.
This happened in large part on 8 May 2007, when Sun contributed the source code to the OpenJDK.

JDK version 7.0 is used.

CLASS DIAGRAMS

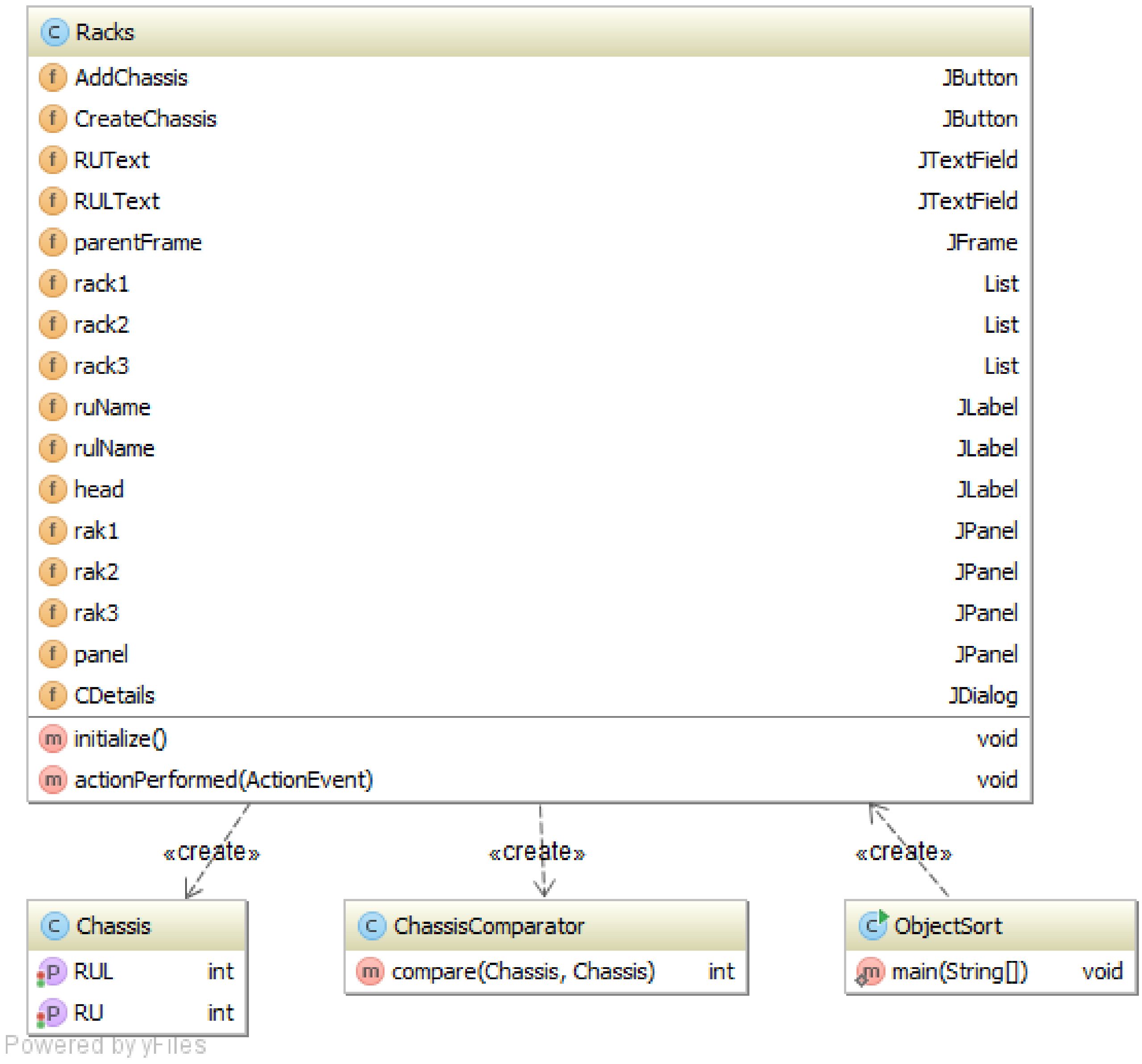
class Chassis			
Data Members	1	RU	int
	2	RUL	
Member Functions	1	getRU	int
	2	getRUL	
	3	setRU(RU:int)	void
	4	setRUL(RUL:int)	

class Racks			
Implementation	1	Implements ActionListener	-
Member Functions	1	Initialize( )	void
	2	actionPerformed(ae:ActionEvent)	void
Data Members	1	AddChassis	JButton
	2	CreateChassis	
	3	RUText	JTextField
	4	RULText	
	5	parentFrame	JFrame
	6	rack1	List
	7	rack2	
	8	rack3	
	9	ruName	JLabel
	10	rulName	
	11	panel	JPanel
	12	CDetails	JDialog

class ChassisComparator			
Implementation	1	Implements Comparator	-
Member Functions	1	Compare(chassis1:Chassis, chassis2:Chassis	-

class ObjectSort			
Function	1	Main(args:String[ ])	void

class Chassis			
Data Members	1	RU	int
	2	RUL	
Member Functions	1	getRU	int
	2	getRUL	
	3	setRU(RU:int)	void
	4	setRUL(RUL:int)	



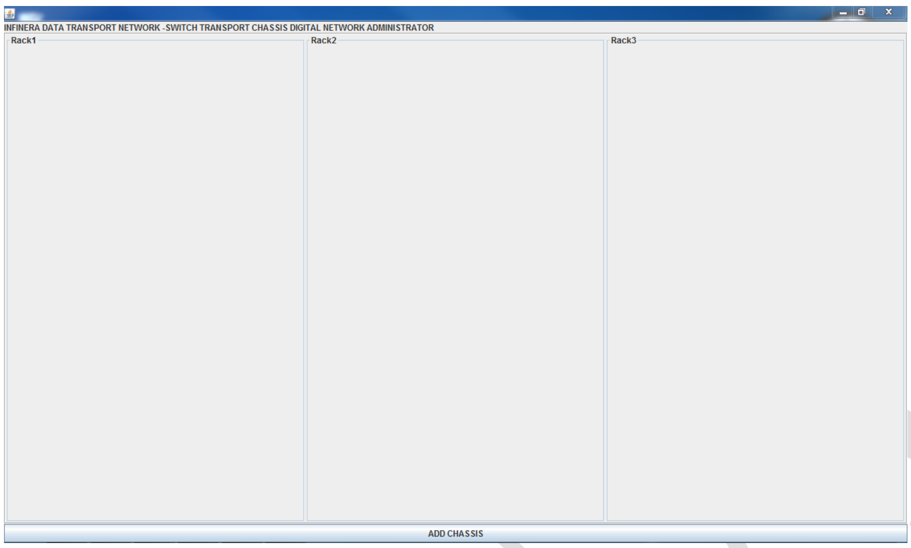
PROJECT CODE

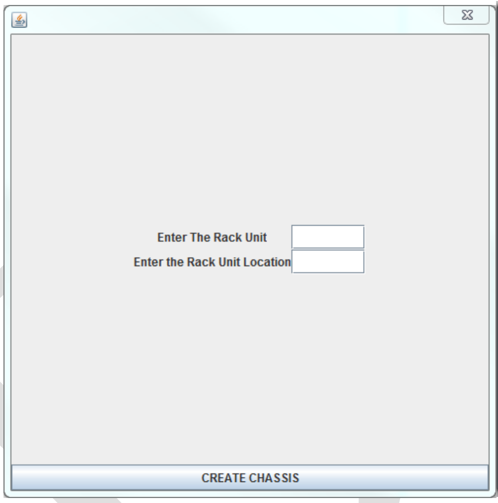
```
import javax.management.relation.RoleUnresolvedList;
import javax.swing.*;
import javax.swing.border.EtchedBorder;
import javax.swing.border.TitledBorder;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.*;
import java.util.List;

class Chassis
{
   int RU;                  //Rack Unit
   int RUL;                 //Rack Unit Location
   Chassis(int RU, int RUL)
   {
      this.RU = RU;
      this.RUL = RUL;
   }
   public int getRU( ) {
      return RU;
   }
   public void setRU(int RU) {
      this.RU = RU;
   }
   public int getRUL( ) {
      return RUL;
   }
   public void setRUL(int RUL) {
      this.RUL = RUL;
   }
}

class ChassisComparator implements Comparator<Chassis>  //Create ChassisComparator and call the comparator function body
{
   public int compare(Chassis chassis1, Chassis chassis2)
   {
      if(chassis1.getRUL( ) < chassis2.getRUL( ))
      {
         return -1;
      }
      else if(chassis1.getRUL( ) == chassis2.getRUL( ))
      {
         return 0;
      }
      else
      {
         return 1;
      }
   }
}
```

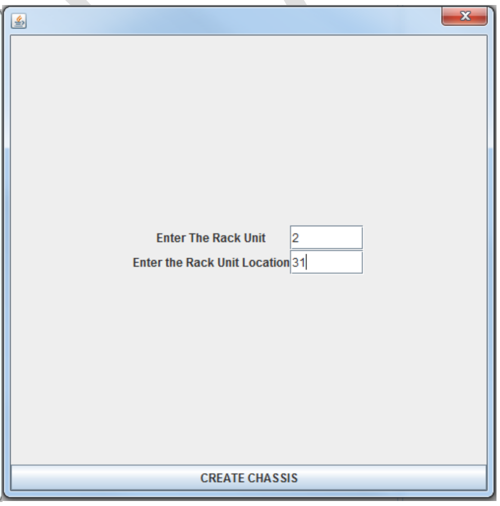
```
class Racks extends JPanel implements ActionListener        //Class Racks declares major components of Java Swing
{
   /* The following declaration is made in class Racks
    * Two Buttons AddChassis and CreateChassis
    * Two Text Fields RUText and RULText -- To accept the Rack Unit (RU) and Rack Unit Location (RUL)
    * Frame parentFrame is created to bear the DNA Chassis View
    * Three Lists rack1, rack2 and rack3
    * Two Labels as titles for the Text Fields
    * Four Panels to hold the chassis
    * Dialog Box CDetails to accept the Rack Unit and Rack Unit Location*/
    JButton AddChassis, CreateChassis;
    JTextField RUText = new JTextField();
    JTextField RULText = new JTextField();
    JFrame parentFrame ;
    List rack1=new ArrayList();
    List rack2=new ArrayList();
    List rack3=new ArrayList();

    JLabel ruName = new JLabel("Enter The Rack Unit");
    JLabel rulName = new JLabel("Enter the Rack Unit Location");
    JLabel head;
    JPanel rak1, rak2, rak3, panel;
    JDialog CDetails;
  /*The above components are initialized in the following constructor
    * Font is modified to SansSerif
    * Give the project a heading
    * Initialize all components*/
    public Racks(JFrame frame)
    {
        final Font f = new Font("SansSerif",Font.CENTER_BASELINE, 50);
        frame.setFont(f);
head = new JLabel("INFINERA DATA TRANSPORT NETWORK -SWITCH TRANSPORT CHASSIS DIGITAL
NETWORK ADMINISTRATOR");

        parentFrame = frame;
        setLayout(new BorderLayout( ));
        RUText.setEditable(true);
        RULText.setEditable(true);
        panel=new JPanel( );
        CDetails = new JDialog(parentFrame);
        initialize();

        RUText.setSize(75,25);
        RUText.setPreferredSize(new Dimension(75, 25));
        RULText.setSize(75,25);
        RULText.setPreferredSize(new Dimension(75, 25));
        rak1=new JPanel();
        rak1.setLayout(new GridLayout(3,1));
        rak1.setBorder(new TitledBorder("Rack1"));

        rak2=new JPanel();
        rak2.setLayout(new GridLayout(3,1));
        rak2.setBorder(new TitledBorder("Rack2"));
        rak3=new JPanel( );
        rak3.setLayout(new GridLayout(3,1));
        rak3.setBorder(new TitledBorder("Rack3"));

        panel.setLayout(new GridLayout(1, 3));
        panel.setLayout(new GridLayout(1, 3));
        panel.setBorder(new EtchedBorder());
        panel.add(rak1);
        panel.add(rak2);
        panel.add(rak3);
        add(panel);
        add(head, BorderLayout.NORTH);
        AddChassis = new JButton("ADD CHASSIS");
        AddChassis.addActionListener(this);
        add(AddChassis, BorderLayout.PAGE_END);
    }

    private void initialize() {
        CDetails.setLayout(new BorderLayout());
        JPanel panel1 = new JPanel();
        panel1.setLayout(new GridBagLayout());
        GridBagConstraints gc = new GridBagConstraints();
        gc.gridx = 0;
        gc.gridy = 0;

        panel1.add(ruName,gc);
        gc.gridx = 1;
        gc.gridy = 0;
        panel1.add(RUText,gc);
        gc.gridx = 0;
        gc.gridy = 1;
        panel1.add(rulName,gc);
        gc.gridx = 1;
        gc.gridy = 1;
        panel1.add(RULText,gc);
        CDetails.add(panel1,BorderLayout.CENTER);
        CreateChassis= new JButton("CREATE CHASSIS");
        CDetails.add(CreateChassis, BorderLayout.PAGE_END);
        CreateChassis.addActionListener(this);
        CDetails.setSize(500, 500);
    }
/* Check for the action of the buttons and performs the necessary tasks like opening, sorting and creating the view.*/
    public void actionPerformed(ActionEvent ae)
    {
        if(ae.getSource( )==AddChassis)
        {
            CDetails.setVisible(true);
        }
        if(ae.getSource( ) == CreateChassis)
        {
            int RU = Integer.parseInt(RUText.getText( ));
            int RUL = Integer.parseInt(RULText.getText( ));
            Chassis chassisObj = new Chassis(RU,RUL);
            GridBagConstraints g = new GridBagConstraints( );
    if(chassisObj.RU == 1)
    {
       rak1.setBackground(Color.BLUE);
       rack1.add(chassisObj);
       int x = rack1.size( );
       rak1.removeAll( );

       Collections.sort(rack1,new ChassisComparator());
       for(int i=0; i<x; i++)
       {
          JButton c1 = new JButton("Rack Unit Location "+((Chassis)rack1.get(i)).RUL+" Rack Unit 1");
          c1.setSize(50,50);
          c1.setBackground(Color.RED);
          g.gridx = 0;
          g.gridy = i;
          rak1.add(c1);
       }
       panel.revalidate( );
       panel.repaint( );
    }
    if(chassisObj.RU == 2)     {
       rak2.setBackground(Color.BLUE);
       rack2.add(chassisObj);
       int x = rack2.size( );
       rak2.removeAll( );

       Collections.sort(rack2,new ChassisComparator( ));
       for(int i=0; i<x; i++)    {
          JButton c1 = new JButton("Rack Unit Location "+((Chassis)rack2.get(i)).RUL+" Rack Unit 2");
          c1.setSize(50,50);
          c1.setBackground(Color.RED);
          g.gridx = 0;
          g.gridy = i;
          rak2.add(c1);
       }
       panel.revalidate( );
       panel.repaint( );
    }
    if(chassisObj.RU == 3)    {
       rak3.setBackground(Color.BLUE);
       rack3.add(chassisObj);
       int x = rack3.size( );
       rak3.removeAll( );

       Collections.sort(rack3,new ChassisComparator( ));
       for(int i=0; i<x; i++)    {
          JButton c1 = new JButton("Rack Unit Location "+((Chassis)rack3.get(i)).RUL+" Rack Unit 3");
          c1.setSize(50,50);
          c1.setBackground(Color.RED);
          g.gridx = 0;
          g.gridy = i;
          rak3.add(c1);
       }
       panel.revalidate( );
       panel.repaint( );
    }
   CDetails.setVisible(false);
  }  }  }
```

```
/* Calls the class Racks */
public class ObjectSort
{
   public static void main(String[] args)
   {
      SwingUtilities.invokeLater(new Runnable( )   {
         @Override
         public void run( ) {
            JFrame frame = new JFrame( );
            Racks calc=new Racks(frame);

           frame. getContentPane( ).add(calc);
            frame.setSize(300,300);

            frame.setVisible(true);
         }
                                             }
    );
   }  }

// END OF PROJECT CODE------------------------------------------------------------
```

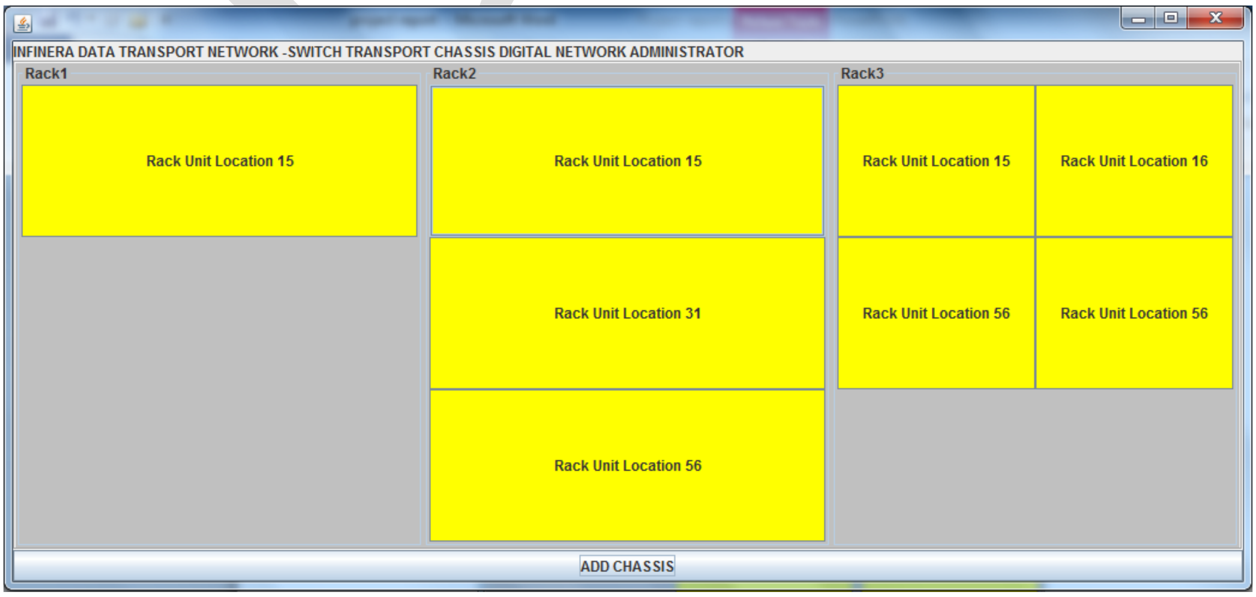
OUTPUT SCREEN SHOTS


INFINERA DATA TRANSPORT NETWORK -SWITCH TRANSPORT CHASSIS DIGITAL NETWORK ADMINISTRATOR
Rack1
Rack2
Rack3
ADD CHASSIS





Enter The Rack Unit 2
Enter the Rack Unit Location 31



SCOPE OF IMPROVEMENT

The Limitations of this project are as follows:

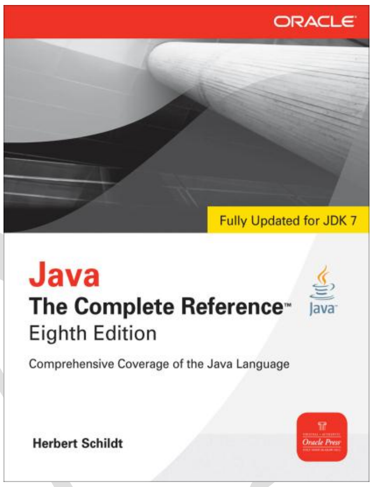
- Number of racks is limited to three racks only.
- If the user enters a rack number beyond 3, then the error message is not displayed.

The rack view of DNA should not be limited to 3 racks. As and when the user enters the rack number, a rack must be created and displayed in the chassis view.

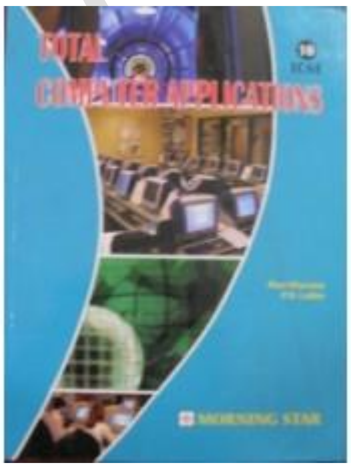
In the current project, since the number of racks is limited, an appropriate error message must be displayed and limit the input to only three racks.

REFERENCES

1. **JAVA : A Complete Reference**
   Eighth Edition
   By Herbert Schildt
   ORACLE Press Publications



2. **Total Computer Applications Part 2 for class 10**
   By Ravi khurana & P.S Latika
   Morning Star Publications



3. **Wikipedia**

4. **Infinera Product Brochure**