

Java Session -2

Variable Scope

► Class Level Scope

```
public class User {  
    private String username;  
}
```

► Method Level Scope

```
public static void main(String[] args) {  
    int x = 5;  
}
```

► Loop Scope

```
public static void main(String[] args) {  
    for (int x = 0; x < 5; x++) {  
        System.out.println("Loop " + x);  
    }  
}
```

ArrayList Container

- ▶ Implements the List interface
 - ▶ Can grow and shrink as needed
 - ▶ Ordered collection (i.e. sequence)
 - ▶ Supports random access
-
- ▶ Online Information
- <https://docs.oracle.com/javase/tutorial/collections/>

ArrayList Container

Instantiation:

```
List<T> list = new ArrayList<>();
```

Method:

```
list.add(val); // add value
```

```
list.get(index); // index: 0 – (length -1)
```

```
list.size(); //
```

```
list.remove(index); // index or object
```

Traverse:

```
for(T t: list) {t.action()}
```

```
For(int i=0;i<list.size();i++) {list.get(i).action()}
```

NullPointerException

- ▶ Null pointer exception is thrown when an application attempts to use null in a case where an object is required. These include:
 - ▶ Calling the instance method of a null object.
 - ▶ Accessing or modifying the field of a null object.
 - ▶ Taking the length of null as if it were an array.
 - ▶ Accessing or modifying the slots of null as if it were an array.
 - ▶ Throwing null as if it were a Throwable value.
 - ▶ Applications should throw instances of this class to indicate other illegal uses of the null object.

Lab2.5 Vital Sign History

The screenshot shows a window titled "Vital Sign History". On the left side, there is a sidebar with the following buttons:

- Create Vital Sign
- View Vital Sign** (highlighted with a blue border)
- MAX_BP 150
- MIN_BP 100
- Abnormal

The main content area has a title "Vital Sign History" and a table showing blood pressure measurements:

Date	Bloodpressure
0911	90.0
0912	110.0

Below the table are four input fields with labels and placeholder text:

- Temperature:
- Bloodpressure:
- Pulse:
- Date:

At the bottom right of the main content area are two buttons: "details" and "delete".

Implement relationship

```
public class VitalSign {  
    private double temperature;  
    private double bloodPressure;  
    private int pulse;  
    private String date;
```

```
private VitalSignHistory vsh;  
/**  
 * Creates new form MainFrame  
 */  
public MainFrame() {  
    initComponents();  
    this.vsh = new VitalSignHistory();  
}
```

```
public class VitalSignHistory {  
    private List<VitalSign> vitalSignList;  
  
    public VitalSignHistory() {  
        this.vitalSignList = new ArrayList<VitalSign>();  
    }  
    public VitalSign addVitalSign(){  
        VitalSign vitalSign = new VitalSign();  
        this.vitalSignList.add(vitalSign);  
        return vitalSign;  
    }  
    public void delVitalSign(VitalSign vitalSign){  
        this.vitalSignList.remove(vitalSign);  
    }  
    public List<VitalSign> getVitalSignList (){  
        return vitalSignList;  
    }  
}
```

Create Vital Sign

Create Vital Sign

Temperature:

Bloodpressure:

Pulse:

Date:

save

```
private void saveBtnActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    double tmp = Double.parseDouble(tmpTextField.getText());  
    double bp = Double.parseDouble(bpTextField.getText());  
    int pulse = Integer.parseInt(pulseTextField.getText());  
    String date = dateTextField.getText();  
    VitalSign vs = vsh.addVitalSign();  
    vs.setTemperature(tmp);  
    vs.setBloodPressure(bp);  
    vs.setPulse(pulse);  
    vs.setDate(date);  
    JOptionPane.showMessageDialog(null,"Vital Sign Created Successfully");  
    resetTxtField();  
}
```

Show vital sign history in JTable

```
private VitalSignHistory vsh;
public ViewPanel(VitalSignHistory vsh) {
    initComponents();
    this.vsh = vsh;
    populateTable();
```



```
private void populateTable(){
    DefaultTableModel dtm = (DefaultTableModel) vitalSignTab.getModel();
    dtm.setRowCount(0);
    for (VitalSign vs : vsh.getVitalSignList()){
        Object row[] = new Object [2];
        row[0] = vs;
        row[1] = vs.getBloodPressure();
        dtm.addRow(row);
    }
}
```

View Detail

```
private void viewDetailActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    int selectedRow = vitalSignTab.getSelectedRow();  
    if (selectedRow>=0){  
        VitalSign vs = (VitalSign) vitalSignTab.getValueAt(selectedRow, 0);  
        bpTextField.setText(String.valueOf(vs.getBloodPressure()));  
        dateTextField.setText(vs.getDate());  
        pulseTextField.setText(String.valueOf(vs.getPulse()));  
        tmpTextField.setText(String.valueOf(vs.getTemperature()));  
    }else {  
        JOptionPane.showMessageDialog(null, "Please select a specific vital sign", "Alert", JOptionPane.ERROR_MESSAGE);  
    }  
}
```

Vital Sign History

Date	Bloodpressure
0911	90.0
0912	110.0

details delete

Temperature: 35.0

Bloodpressure: 90.0

Pulse: 90

Date: 0911

Vital Sign History

Delete vital sign

```
private void deleteActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    int selectedRow = vitalSignTab.getSelectedRow();  
    if (selectedRow >= 0){  
        int dialogRes = JOptionPane.showConfirmDialog(null, "This vital sign  
        if(dialogRes == JOptionPane.YES_OPTION){  
            VitalSign vs = (VitalSign) vitalSignTab.getValueAt(selectedRow, 0);  
            vsh.delVitalSign(vs);  
            populateTable();  
            resetTextField();  
        }  
    }else{  
        JOptionPane.showMessageDialog(null, "Please select a specific vital sign to delete", "  
    }
```

Date	Bloodpressure
0911	90.0
0912	110.0

[details](#) [delete](#)

Temperature:	35.0
Bloodpressure:	90.0
Pulse:	90
Date:	0911

Find the Abnormal Vital Signs

Abnormal vital sign

Date	Bloodpressure
0911	90.0

details **delete**

Temperature:

Bloodpressure:

Pulse:

Date:

Create Vital Sign

View Vital Sign

MAX_BP 150

MIN_BP 100

Abnormal

```
private void abnBtnActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    String maxbpString = maxTxt.getText();  
    String minbpString = minTxt.getText();  
    double maxbp = maxbpString.equals("") ? Double.MAX_VALUE : Double.parseDouble(maxbpString);  
    double minbp = minbpString.equals("") ? Double.MIN_VALUE : Double.parseDouble(minbpString);  
    AbnormalPanel abnormalPanel = new AbnormalPanel(vsh, maxbp, minbp);  
    jSplitPanel.setRightComponent(abnormalPanel);  
}
```

Main JFrame

```
private void populateTable() {  
    DefaultTableModel dtm = (DefaultTableModel) vitalSignTab.getModel();  
    dtm.setRowCount(0);  
    for (VitalSign vs : vsh.getAbnormalList(maxbp, minbp)) {  
        Object row[] = new Object[2];  
        row[0] = vs;  
        row[1] = vs.getBloodPressure();  
        dtm.addRow(row);  
    }  
}
```

Abnormal JPanel

```
public List<VitalSign> getAbnormalList(double maxbp, double minbp){  
    List<VitalSign> abnList = new ArrayList<>();  
  
    for(VitalSign vs:vitalSignList){  
        if(vs.getBloodPressure()>maxbp || vs.getBloodPressure()<minbp){  
            abnList.add(vs);  
        }  
    }  
    return abnList;  
}
```

VitalsignHistory

Homework

- ▶ Finish the vital sign application according to the video (lab2.5)
- ▶ Bonus:(5 points)
try to implement the functionality of finding the abnormal vital signs and display them.

for example: blood pressure $> 140 \mid\mid$ blood pressure < 70 is abnormal.

You can have your own criteria.