

Directory.java

Your job is to fill in the missing code in Directory.java. The comments, method and variable names should give you enough context to be able to fill in the blanks. You only need to fill in where there are blanks.

Directory.java is a program whose purpose is to check membership within a group.

The three basic operations of Directory.java are

- enter – adds a name, id pair to the group
- remove – removes a name from the group and its corresponding id
- lookup – determines if a name is in the group. If the name is in the group, then the id field is set with the user's id and the message field indicates that the name is in the group. Otherwise, the message field indicates the name is not in the group.



The screenshot shows a window titled "Access Control". It contains three input fields: "Name:", "ID:", and "Message:". The "Message:" field contains the text "Jarn added as a member.". Below the input fields are four buttons: "remove", "enter", "lookup", and "quit".

Directory.java

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

/*
 * Created on Feb 28, 2004
 */

/**
 * @author Jam Jenkins
 *
 * This class simulates a directory of students and ids
 */
public class Directory extends JApplet implements ActionListener {

    private JFrame frame;
    private JButton removeButton, enterButton,
lookupButton, quitButton;
    private JLabel nameLabel, idLabel, messageLabel;
    private JTextField nameField, idField, messageField;
    private HashMap members;

    public Directory(){
        super();
        makeComponents();
        layoutComponents();
        members=new HashMap();
    }
}
```

Directory.java

```
/** initialize all of the component instance
 * variables and add this as an
 * ActionListener to all JButtons*/
private void makeComponents()
{
    frame=new JFrame("Access Control");
    removeButton=new JButton("remove");
    removeButton.addActionListener(this);
    _____;
    _____;
    _____;
    _____;
    _____;
    _____;
    nameLabel=new JLabel("Name:");
    _____;
    _____;
    nameField=new JTextField(20);
    _____;
    _____;
}
}
```

Directory.java

```
/**put all components in the JFrame's container*/
private void layoutComponents()
{
    Container container=frame.getContentPane();
    container.setLayout(_____);
    container.add(_____);
    container.add(_____);
    frame.pack();
}

/**
 * @return the JPanel with all the JLabels and JTextFields
 */
private JPanel getMiddlePanel()
{
    _____;
    panel.add(nameLabel);
    panel.add(nameField);
    panel.add(idLabel);
    panel.add(idField);
    panel.add(messageLabel);
    panel.add(messageField);
    return panel;
}
```

Directory.java

```
/**
 * @return the JPanel with all the JButtons
 */
private JPanel getButtonPanel()
{
    JPanel panel=new JPanel(new GridLayout(1, 4));
    panel.add(_____);
    panel.add(_____);
    panel.add(_____);
    panel.add(_____);
    return panel;
}
```

Directory.java

```
/**
 * removes a user to members by getting input
 * from nameField. If the user existed, the
 * messageField indicates the user was removed,
 * otherwise the messageField indicates that no
 * such user was a member
 */
private void remove(){
    String name=nameField.getText();
    if(members.containsKey(name)){
        members.remove(name);
        messageField.setText(name+" removed from members.");
    }
    else{
        messageField.setText(name+" is not a member.");
    }
}

/**
 * adds a user to members by getting input
 * from nameField and idField
 */
private void enter(){
    String name=nameField.getText();
    String id=idField.getText();
    members.put(name, id);
    messageField.setText(name+" added as a member.");
    nameField.setText("");
    idField.setText("");
}
}
```

```
/**
 * looks up a user by name and outputs the
 * user id in the idField (if found) and
 * indicates whether the user is or is not
 * a member in the messageField
 */
private void lookup()
{
    String name=nameField.getText();
    if(members.containsKey(name)){
        String id=(String)members.get(name);
        idField.setText(id);
        messageField.setText(name+" is a member.");
    }
    else{
        messageField.setText(name+" is not a member.");
    }
}

/**
 * quits the program
 */
private void quit(){
    frame.setVisible(false);
    frame.dispose();
    System.exit(0);
}
}
```

Directory.java

```
public void actionPerformed(ActionEvent e) {
    Object cause=e.getSource();
    if(cause==_____){
        //removes a user and corresponding id
        remove();
    }
    else if(cause==enterButton){
        //adds a user and corresponding id
        _____;
    }
    else if(_____) {
        //looks up a user and corresponding id
        _____;
    }
    else if(_____) {
        //quits the program
        _____;
    }
}
```

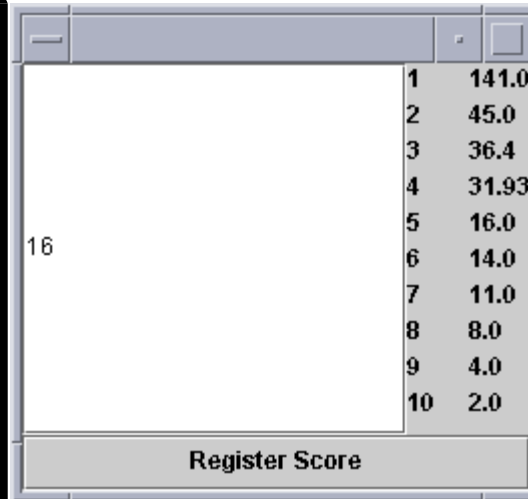
Directory.java

```
public void init(){
    frame.setVisible(true);
}

public static void main(String[] args) {
    Directory directory=new Directory();
    directory.init();
}
}
```

TopTen

Your mission is to fill in the missing code in TopTen.java, TopTenPanel.java, and TopTenTest.java. The comments, method and variable names should give you enough context to be able to fill in the blanks. You only need to fill in where there are blanks. A description of each class is given to the right.



A screenshot of a Java Swing window titled "Register Score". The window contains a list of 10 scores, numbered 1 through 10, displayed in a table format. The scores are: 141.0, 45.0, 36.4, 31.93, 16.0, 14.0, 11.0, 8.0, 4.0, and 2.0. The scores are sorted in descending order. The window has a standard Mac OS-style title bar with a close button on the right. A small number "16" is visible on the left side of the window's content area.

1	141.0
2	45.0
3	36.4
4	31.93
5	16.0
6	14.0
7	11.0
8	8.0
9	4.0
10	2.0

- TopTen.java – this class keeps the scores in an array of doubles sorted least to greatest. It extends Observable and is the model.
- TopTenPanel – this class displays the scores in a JPanel which it extends. This class is the view.
- TopTenTest – this class is the controller and it connects the model (TopTen) and the view (TopTenPanel) in order to test them.

TopTen.java

```
import java.util.*;

/**
 * @author Jam Jenkins
 *
 * This class keeps a list of the top ten scores.
 */
public class TopTen extends Observable{
    /**scores is ordered in ascending order*/
    double[] scores;

    public TopTen()
    {
        //initialize score to hold 10 doubles
        _____;
    }
}
```

TopTen.java

```
/**
 * add a score if it ranks in the top ten
 * @param s the score to add
 */
public void registerScore(double s)
{
    //remember that score is sorted least to greatest
    if(_____)//score is better than the lowest score
    {
        scores[0]=s;
        Arrays.sort(scores);
        setChanged();
        notifyObservers();
    }
}
```

TopTen.java

```
/**
 * get a score from the top ten
 * @param rank the position in the top ten list
 * with 1 being the highest and 10 being the lowest
 * @return the score
 */
public double getScore(int rank)
{
    //score is sorted least to greatest
    //think about how the rank relates to the index in the array
    return _____;
}
}
```

TicTacToe.java

```
import java.awt.*;
import javax.swing.*;
import java.util.*;
/**
 * @author Jam Jenkins
 *
 * Displays the scores in a JPanel
 */
public class TopTenPanel extends JPanel
    implements Observer{
    /**JLabels with the numbers 1-10*/
    private JLabel[] ranks;
    /**the numeric scores for the top ten*/
    private JLabel[] scores;

    public TopTenPanel()
    {
        super();
        makeComponents();
        layoutComponents();
    }
}
```

TopTenPanel.java

```
private void makeComponents()
{
    ranks=new JLabel[10];
    scores=new JLabel[10];
    for(int i=0; _____; i++)
    {
        ranks[i]=new JLabel(""+(i+1));
        scores[i]=new JLabel("none");
    }
}

private void layoutComponents()
{
    setLayout(new GridLayout(10, 2));
    for( _____; _____; _____ )
    {
        add( _____ );
        add( _____ );
    }
}
```

TopTenPanel.java

```
/**
 * resolve differences in the TopTen and what
 * is currently in scores by setting the text
 * of the scores array
 *
 * @param o the TopTen
 * @param arg null
 */
public void update(Observable o, Object arg)
{
    TopTen ten=(TopTen)o;
    for(int i=0; i<10; i++)
    {
        //use i+1 since rank goes 1-10 and indexes go 0-9
        _____ .setText(" "+ten.getScore(i+1));
    }
}
}
```

TopTenTest.java

```
/*
 * Created on Feb 28, 2004
 */
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

/**
 * @author Jam Jenkins
 *
 * Test the classes TopTen and TopTenPanel
 *
 */
public class TopTenTest extends JApplet
    implements ActionListener
{
    private TopTen ten;
    private TopTenPanel tenPanel;
    private JFrame frame;
    private JTextField scoreField;
    private JButton enterButton;
```

TopTenTest.java

```
public TopTenTest()  
{  
    super();  
    makeComponents();  
    layoutComponents();  
    ten.addObserver(tenPanel);  
}  
  
private void makeComponents()  
{  
    ten=new TopTen();  
    tenPanel=new TopTenPanel();  
    frame=new JFrame();  
    scoreField=new JTextField(5);  
    enterButton=new JButton("Register Score");  
    enterButton.addActionListener(this);  
}
```

TopTenTest.java

```
private void layoutComponents()  
{  
    Container container=frame.getContentPane();  
    container.setLayout(______);  
    container.add(scoreField,______);  
    container.add(enterButton, ______);  
    container.add(tenPanel,______);  
}  
  
public void actionPerformed(ActionEvent e)  
{  
    String scoreText=scoreField.getText();  
    double score=Double.parseDouble(scoreText);  
    ten.registerScore(score);  
}
```

TopTenTest.java

```
public void init()  
{  
    frame.pack();  
    frame.setVisible(true);  
}  
  
public static void main(String[] args)  
{  
    TopTenTest test=new TopTenTest();  
    test.init();  
}  
}
```

Collections

- Why is HashMap a better choice than ArrayList for the collection to use in Directory.java?
- Why is an array a bad choice for Directory.java but a good choice for TopTen.java?
- What is one benefit of using an array as opposed to an ArrayList?
- What is one benefit of using an ArrayList as opposed to an array?
- What would be a reason not to use a TreeMap to hold the scores in TopTen.java?

Searching

- What is the primary benefit of linear (sequential) search over binary search?
- What is the primary benefit of binary search over linear (sequential) search?
- When is the speed of binary search and linear search most dramatically different?

Tracing

Show the output that results from executing the code on the right

```
int sum=0;
for(int i=0; i<10; i++)
{
    if(i%2==1)
    {
        sum+=i;
    }
}
System.out.println("sum is "+sum)
sum=6;
int target=3;
if(sum<target)
{
    System.out.println("less");
}
if(sum>target)
{
    System.out.println("greater");
}
if(sum!=target)
{
    System.out.println("not equal");
}
```