단국대학교 응용컴퓨터공학 JAVA 프로그래밍 I 기말고사 (2017년 가을학기) 2017년 12월 15일

학과 _____ 학번 _____ 이름 _____

기말고사

담당교수: 단국대학교 응용컴퓨터공학 박경신

- **답은 반드시 답안지에 기술할 것. 공간이 부족할 경우 반드시 답안지 몇 쪽의 뒤에 있다고 명기한 후 기술할 것. 그 외의 경우의 답안지 뒤쪽이나 연습지에 기술한 내용은 답안으로 인정 안 함. 답에는 반드시 네모를 쳐서 확실히 표시할 것.**
- **답안지에 학과, 학번, 이름 외에 본인의 암호(4자리 숫자)를 기입하면 성적공고시 학번대신 암호를 사용할 것임.**

```
// ArithmeticOperator
package ValueLib;
public enum ArithmeticOperator {
	PLUS,
	MINUS,
	TIMES,
	DIVIDE,
	REMAINDER,
	POWER;
	public int calculate(int x, int y) { // (1)
		switch (this) {
			case PLUS:      return x + y;
			case MINUS:     return x - y;
			case TIMES:     return x * y;
			case DIVIDE:    return x / y;
			case REMAINDER: return x % y;
			case POWER:     return (int)Math.pow(x, y);
			default: throw new AssertionError("Unknown operations " + this);
		}
	}
	public String getOperator() { // (1)
		switch (this) {
			case PLUS:      return "+";
			case MINUS:     return "-";
			case TIMES:     return "*";
			case DIVIDE:    return "/";
			case REMAINDER: return "%";
			case POWER:     return "^";
			default: throw new AssertionError("Unknown operations " + this);
		}
	}
	public static ArithmeticOperator valueOf(int value) { // (1)
		switch (value) { // switch
			case 0:         return PLUS;
			case 1:         return MINUS;
			case 2:         return TIMES;
			case 3:         return DIVIDE;
			case 4:         return REMAINDER;
			case 5:         return POWER;
		}
		return null;
	}
	public static String[] names() { // (1)
		return new String[] {"+", "-", "*", "/", "%", "^"};
	}
```

학과 _____ 학번 _____ 이름 _____

```
	public static ArithmeticOperator nameOf(String name) { // (1)
		switch (name) { // switch
			case "+":	return PLUS;
			case "-":	return MINUS;
			case "*":	return TIMES;
			case "/":	return DIVIDE;
			case "%":	return REMAINDER;
			case "^":	return POWER;
		}
		return null;
	}
}

// Value
package ValueLib;
import java.util.*;
public class Value implements Comparable<Value> { // (3)
	protected int a;
	protected int b;
	public Value() {
		this(0, 0);
	}
	public Value(int[] value) {
		this(value[0], value[1]);
	}
	public Value(int a, int b) {
		set(a, b);
	}
	public void set(int a, int b) { // (2)
		this.a = a;
		this.b = b;
	}
	public int getA() {
		return this.a;
	}
	public int getB() {
		return this.b;
	}
	public void setA(int a) {
		this.a = a;
	}
	public void setB(int b) {
		this.b = b;
	}
	@Override
	public String toString() { // (2)
		return a + "," + b;
	}
	@Override
	public int compareTo(Value other) { // (4)
		return Integer.compare(this.a, other.getA());
	}
	public static Comparator<Value> BComparator = new Comparator<Value>() {
		public int compare(Value v1, Value v2) {	// (4)
			return Integer.compare(v1.getB(), v2.getB());
		}
	};
}
```

학과 _____ 학번 _____ 이름 _____

```
// ValueCalculator
package ValueLib;
public class ValueCalculator extends Value { // (3)
	protected ArithmeticOperator op;
	protected int c;
	public ValueCalculator() {
		this(0, 0, ArithmeticOperator.PLUS);
	}
	public ValueCalculator(int a, int b, ArithmeticOperator op) {
		set(a, b, op);
	}
	public ValueCalculator(int a, int b, ArithmeticOperator op, int c) {
		super(a, b);
		this.op = op;
		this.c = c;
	}
	public void set(int a, int b, ArithmeticOperator op) { // (2)
		this.a = a;
		this.b = b;
		this.op = op;
		this.c = op.calculate(a, b);
	}
	public ArithmeticOperator getOp() {
		return op;
	}
	public int getC() {
		return c;
	}
	@Override
	public String toString() { // (2)
		return this.a + "," + this.b + "," + this.op + "," + this.c;
	}
}

// ValueCalculatorComparator
package ValueLib;
import java.util.*;
public class ValueCalculatorComparator implements Comparator<ValueCalculator> { // (4)
	private int column = 0;
	public ValueCalculatorComparator() {
		this(0);
	}
	public ValueCalculatorComparator(int column) {
		this.column = column;
	}
	@Override
	public int compare(ValueCalculator c1, ValueCalculator c2) {
		if (!(c1 instanceof ValueCalculator) || !(c2 instanceof ValueCalculator))
			return 0;
		switch(this.column) {
			case 0: return Integer.compare(c1.getA(), c2.getA());
			case 1: return Integer.compare(c1.getB(), c2.getB());
			case 2: return c1.getOp().compareTo(c2.getOp());
			case 3: return Integer.compare(c1.getC(), c2.getC());
			default: return 0;
		}
	}
}
```

학과 _____ 학번 _____ 이름 _____

```java
// Utility
package ValueLib;
public class Utility {
	public static int tryParseInt(String str) { // (5) String -> int
		int val = 0;
		try {
			val = Integer.parseInt(str);
		} catch (NumberFormatException e) {
			e.printStackTrace();
		}
		return val;
	}
}
```

```java
// ValueImporter
package ValueLib;
import java.io.*;
import java.util.*;
public class ValueImporter {
	public static List<Value> loadCSV(String filename) {
		List<Value> valueList = new ArrayList<Value>();
		try {
			BufferedReader in = new BufferedReader(new FileReader(filename)); //(6)
			String line;
			String delimiter = ",";
			while ((line = in.readLine()) != null) {
				String[] items = line.split(delimiter);
				int a = Utility.tryParseInt(items[0]); // (5)
				int b = Utility.tryParseInt(items[1]); // (5)
				Value v = new Value(a, b);
				valueList.add(v);
			}
			in.close();
		} catch (IOException e) {
			e.printStackTrace();
			System.exit(1);
		} catch (Exception e) {
			e.printStackTrace();
		}
		return valueList;
	}
	public static void writeCSV(String path, List<Value> valueList) {
		FileWriter fw; // (6)
		try {
			fw = new FileWriter(path);
			for (Value v : valueList) {
				fw.append(v.toString() + "\n");
			}
			fw.flush();
			fw.close();
		} catch (IOException e) {
			e.printStackTrace();
		}
	}
}
```

학과 _____ 학번 _____ 이름 _____

```
// ValueCalculatorManager
package ValueLib;
import java.util.*;
import java.util.function.Predicate;
import java.util.stream.Collectors;
public class ValueCalculatorManager {
	private List<ValueCalculator> list = null;
	public ValueCalculatorManager() {
		this.list = new ArrayList<ValueCalculator>();
	}
	public ValueCalculatorManager(List<ValueCalculator> list) {
		this.list = list;
	}
	public void add(ValueCalculator value) {
		this.list.add(value);
	}
	public int size() {
		return this.list.size();
	}
	public boolean isEmpty() {
		return this.list.isEmpty();
	}
	public ValueCalculator getAt(int index) {
		return this.list.get(index);
	}
	public void sort(int index) {
		this.list.sort(new ValueCalculatorComparator(index));
	}
	@Override
	public String toString() {
		return java.util.Arrays.toString(this.list.toArray());
	}
	public void load(String filename) { // (7)
		List<Value> vList = ValueImporter.loadCSV(filename);
		for (Value v : vList) {
			for (ArithmeticOperator op : ArithmeticOperator.values()) {
				add(new ValueCalculator(v.getA(), v.getB(), op));
			}
		}
	}
}

// ValueCalculatorTableModel
package ValueFrame;
import javax.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.table.*;
import ValueLib.ValueCalculator;
import ValueLib.ValueCalculatorManager;
public class ValueCalculatorTableModel extends AbstractTableModel {
	private ValueCalculatorManager manager;
	private String[] columnNames = { "A", "B", "OP", "C"};
	public ValueCalculatorTableModel(ValueCalculatorManager manager) {
		this.manager = manager;
	}
	@Override
	public int getRowCount() {
		return this.manager.size();
	}
	@Override
```

학과 _____ 학번 _____ 이름 _____

```
	public int getColumnCount() {
		return columnNames.length;
	}
	@Override
	public String getColumnName(int columnIndex) {
		return columnNames[columnIndex];
	}
	@Override
	public Class<?> getColumnClass(int columnIndex) {
		if (this.manager.isEmpty()) {
			return Object.class;
		}
		return getValueAt(0, columnIndex).getClass();
	}
	@Override
	public Object getValueAt(int rowIndex, int columnIndex) {
		ValueCalculator calc = this.manager.getAt(rowIndex);
		Object value = null;
		switch (columnIndex) {
			case 0:
				value = calc.getA();
				break;
			case 1:
				value = calc.getB();
				break;
			case 2:
				value = calc.getOp().getOperator();
				break;
			case 3:
				value = calc.getC();
				break;
		}
		return value;
	}
	public void addRow(ValueCalculator v) {
		this.manager.add(v);
		fireTableRowsInserted(this.manager.size() - 1, this.manager.size() - 1);
	}
	class ColumnListener extends MouseAdapter {
		private JTable table;
		public ColumnListener(JTable table) {
			this.table = table;
		}
		public void mouseClicked(MouseEvent e) {
			TableColumnModel colModel = table.getColumnModel();
			int columnModelIndex = colModel.getColumnIndexAtX(e.getX());
			int modelIndex = colModel.getColumn(columnModelIndex).getModelIndex();
			if (modelIndex < 0) return;
			for (int i = 0; i < getColumnCount(); i++) {
				TableColumn column = colModel.getColumn(i);
				column.setHeaderValue(getColumnName(column.getModelIndex()));
			}
			table.getTableHeader().repaint();
			manager.sort(columnModelIndex);
			table.tableChanged(new TableModelEvent(ValueCalculatorTableModel.this));
			table.repaint();
		}
	}
}
```

학과 _____ 학번 _____ 이름 _____

```
// ValueCalculatorTableFrame
package ValueFrame;
import javax.swing.*;
import java.awt.*;
import javax.swing.table.*;
import ValueLib.ValueCalculator;
import ValueLib.ValueCalculatorManager;
public class ValueCalculatorTableFrame extends JFrame {
        public ValueCalculatorTableModel model = null;
        public ValueCalculatorTableFrame(ValueCalculatorManager manager) {
                super("ValueCalculatorTableFrame");
                this.model = new ValueCalculatorTableModel(manager);
                JTable table = new JTable(this.model);
                table.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
                table.setIntercellSpacing(new Dimension(10, 5));
                JTableHeader header = table.getTableHeader();
                header.setUpdateTableInRealTime(true);
                header.addMouseListener(model.new ColumnListener(table)); // (8)
                header.setReorderingAllowed(true);
                JScrollPane scrollPane = new JScrollPane(table);
                this.add(scrollPane, BorderLayout.CENTER);
                this.setSize(600, 600);
                this.setVisible(true);
                this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
}

// ValueCalculatorFrame
package ValueFrame;
import java.awt.event.*;
import javax.swing.*;
import ValueLib.ArithmeticOperator;
import ValueLib.Utility;
import ValueLib.ValueCalculator;
public class ValueCalculatorFrame extends JFrame implements ActionListener {
        JLabel label1 = new JLabel("A");
        JLabel label2 = new JLabel("B");
        JLabel label3 = new JLabel("연산자");
        JTextField textfield1 = new JTextField(20);
        JTextField textfield2 = new JTextField(20);
        JComboBox<String> combobox1 = new JComboBox<String>(ArithmeticOperator.names()); //(9)
        JButton button1 = new JButton("계산");
        ValueCalculatorTableFrame tableFrame = null;
        public ValueCalculatorFrame(ValueCalculatorTableFrame tableFrame) {
                setTitle("ValueCalculatorFrame");
                setLayout(null); // no layout
                this.tableFrame = tableFrame;
                label1.setBounds(20, 30, 100, 20);
                label2.setBounds(20, 60, 100, 20);
                label3.setBounds(20, 90, 100, 20);
                this.add(label1);
                this.add(label2);
                this.add(label3);
                textfield1.setBounds(120, 30, 150, 20);
                textfield2.setBounds(120, 60, 150, 20);
                this.add(textfield1);
                this.add(textfield2);
                combobox1.setBounds(120, 90, 150, 20);
                this.add(combobox1);
                button1.setBounds(60, 120, 160, 20);
                button1.addActionListener(this); // (9)
```

```
                this.add(button1);
                setSize(300, 200);
                this.setVisible(true);
                this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
        public ArithmeticOperator select() {
                return ArithmeticOperator.valueOf(combobox1.getSelectedIndex()); // (9)
        }
        @Override
        public void actionPerformed(ActionEvent e) {
                if ((JButton)e.getSource() == button1) {
                        ValueCalculator calc = new ValueCalculator();
                        int a = Utility.tryParseInt(textfield1.getText()); // (9)
                        int b = Utility.tryParseInt(textfield2.getText()); // (9)
                        ArithmeticOperator op = select();
                        calc.set(a, b, op);
                        tableFrame.model.addRow(calc);
                }
        }
}

// MainFrame
package ValueFrame; // (10)
import ValueLib.ValueCalculatorManager; // (10)
public class MainFrame {
        public static void main(String args[]) {
                ValueCalculatorManager manager = new ValueCalculatorManager();
                manager.load("valuelist.csv"); // (7)
                ValueCalculatorTableFrame tableFrame = new ValueCalculatorTableFrame(manager);
                new ValueCalculatorFrame(tableFrame);
        }
}
```

"valuelist.csv"

10,6

8,4

9,2

7,5

8,3

단국대학교 응용컴퓨터공학 JAVA 프로그래밍 I 기말고사 (2017년 가을학기) 2017년 12월 15일

학과 _____ 학번 _____ 이름 _____

1. ArithmeticOperator 코드의 빈칸을 채워라. (10점)

2. Method Overloading과 Method Overriding이 무엇인지 자세히 설명하라. Value와 ValueCalculator 클래스에서 예시를 찾아서 적어라. (10점)

메소드 오버로딩은 동일한 메소드 이름에 매개변수가 다른 함수를 둘 이상 정의하는 것으로,
동일한 함수 기능을 수행하지만 다른 매개변수의 경우를 처리할 때 사용
예로, Value 클래스 void set(int, int)와 ValueCalculator 클래스 void set(int, int, ArithmeticOperator)

메소드 오버라이딩은 상속관계에서 상속받은 파생클래스에서 동일한 메소드 이름에
동일한 매개변수와 반환자로 정의하여 함수를 재정의하는 것으로 상속되어진 함수의 기능을
변경해서 재사용하고 싶을 때 사용
예로, Value 클래스와 ValueCalculator 클래스의 String toString() 메소드는 Object 클래스의
toString()를 재정의해서 사용

3. implements 와 extends가 무엇인지 예를 들어 자세히 설명하라. (5점)

class Value implements Comparable<Value>는 Comparable 인터페이스를 상속받아서 해당 인터페이스의
메소드 즉 int compareTo(Value)를 구현하는 것이다.

class ValueCalculator extends Value는 ValueCalculator 클래스가 Value 클래스를 상속받아서
확장한다는 것이다.
즉, Value 클래스의 protected와 public 필드와 메소드를 상속받아서 사용가능하고
추가적인 메소드를 더 구현하는 것이다.

4. Value 클래스와 ValueCalculatorComparator 클래스 코드의 빈칸을 채워라. (15점)

5. Utility 클래스의 int tryParseInt(String) 메소드를 try/catch를 사용하여 구현하라. (5점)

6. ValueImporter 클래스에서 사용한 FileReader, FileWriter, BufferedReader 클래스를 설명하라. (5점)

FileReader와 FileWriter 클래스는 문자 데이터 파일을 읽고 처리하는 클래스이다.
BufferedReader는 문자 데이터 파일을 읽고 처리하는 버퍼스트림 클래스이다.
버퍼를 가진 입출력 클래스들은 입출력 데이터를 일시적으로 저장하는 버퍼를 이용하여 입출력 효율을
높이려고 한다.

7. ValueCalculatorManager 클래스 void load(String) 메소드의 동작 원리를 설명하라. (10점)

MainFrame 클래스의 메인 메소드에서 manager.load("valuelist.csv")에서 호출
ValueImporter.loadCSV를 이용하여, 파일을 읽어 들여 List<Value> vList로 받는다.
중첩 foreach 즉, vList에 있는 모든 Value와 모든 ArithmeticOperator 마다 ValueCalculator 연산을 한다.
그리고, 그 ValueCalculator를 List<ValueCalculator>에 add 한다.

학과 _____ 학번 _____ 이름 _____

8. ValueCalculatorTableFrame 클래스에서 사용한 addMouseListener 메소드의 동작 원리를 설명하라. 만약 이 메소드를 호출하지 않았을 경우 (즉, comment out 했을 시) 실행결과가 어떻게 바뀌는지 자세히 설명하라. (10점)

addMouseListener는 MouseListener 이벤트 핸들러 메소드를 등록하는 것으로
ValueCalculatorTableModel 클래스의 내부 클래스인 ColumnListener 클래스에
mouseClicked(MouseEvent e)를 구현하였다.

만약 addMouseListener를 comment out 해서 호출하지 않았을 경우,
테이블 컬럼 헤더를 마우스로 클릭해도, 그 해당 컬럼(즉, A, B, OP, C)으로 sort가 되지 않게 된다.

9. ValueCalculatorFrame 클래스 코드의 빈칸을 채워라. (10점)

10. package와 import가 무엇인지 예를 들어 자세히 설명하라. (10점)

Package는 서로 관련된 클래스와 인터페이스 등을 하나의 디렉토리에 묶어 놓은 것
하나의 응용프로그램은 여러 개의 패키지로 작성 가능하다.
예를 들어, ValueLib 패키지에는 ArithmeticOperator, Value, ValueCalculator, ValueCalculatorComparator, Utility, ValueImporter, ValueCalculatorManager를 묶었고,
ValueFrame 패키지에는 ValueCalculatorTableModel, ValueCalculatorTableFrame, ValueCalculatorFrame, MainFrame을 묶어서 사용하였다.

Import는 외부 클래스를 사용하여 구현할 때, 해당 클래스가 속해있는 절대 경로명
(즉, 패키지명과 클래스 이름 또는 패키지명.*)을 호출해야 사용가능하다.
Import를 사용하여 소스의 시작부분에 사용하려는 클래스의 패키지를 명시함.

11. 이 응용프로그램의 동작 원리 (즉, 실행 결과)를 자세히 설명하라. (10점)

이 응용프로그램은 MainFrame 클래스의 메인 메소드에서 manager.load("valuelist.csv")에서 호출하여
5개의 Value에 대한 6개의 ArithmeticOperator 연산을 한 ValueCalculator 값이
테이블(ValueCalculatorTableFrame)에 출력된다.

테이블의 컬럼헤더를 마우스로 클릭하면, 해당 컬럼헤더의 값으로 sort가 된다.

그리고, ValueCalculatorFrame에서는 A, B에 텍스트로 값을 입력하고 콤보박스에서 Operator를 선택하여
"계산" 버튼을 누르면, ValueCalculator 연산한 값이 테이블에 추가된다.

-끝-