단국대학교 응용컴퓨터공학 JAVA 프로그래밍 I 기말고사 (2018년 봄학기) 2018년 6월 12일

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.**

1. 다음 sub1과 sub2의 실행결과가 같도록, sublist 메소드 내부의 코드를 완성하라. (10점)

```
public static int[] sublist(int[] list, int fromIndex, int toIndex) {
    int size = toIndex - fromIndex;
    int[] result = new int[size];
    for (int i = 0; i < size; i++) {
        result[i] = list[fromIndex + i];
    }
    return result;



}
```

```
public static List<Integer> sublist(List<Integer> list, int fromIndex, int toIndex) {
    List<Integer> result = new ArrayList<Integer>();
    for (int i = fromIndex; i < toIndex; i++) {
        result.add(list.get(i));
    }
    return result;



}
```

```
public static void main(String[] args) {
    int[] dataArray = {7, 5, 8, 5, 9, 7, 2, 3};
    int[] sub1 = sublist(dataArray, 2, 5);
    System.out.println("sub1 = " + Arrays.toString(sub1)); // sub1 = [8, 5, 9]

    List<Integer> dataList = Arrays.asList(7, 5, 8, 5, 9, 7, 2, 3);
    List<Integer> sub2 = sublist(dataList, 2, 5);
    System.out.println("sub2 = " + Arrays.toString(sub2.toArray())); // sub2 = [8, 5, 9]
}
```

학과 _____ 학번 _____ 이름 _____

2. 다음 코드의 실행 결과를 적어라 (parameter passing에 유의할 것). (10점)

```
public static int mystery(int n, int[] numbers) {
    n += 100;
    numbers[2]--;
    System.out.println(n + " " + Arrays.toString(numbers));
    return numbers[1] * 2;
}

public static void mystery(int[] x, int[] y) {
    int[] t = x;
    x = y;
    y = t;
    System.out.println(Arrays.toString(x) + " " + Arrays.toString(y));
}

public static void mystery(int[] list) {
    for (int i = 1; i < list.length - 1; i++) {
        if (list[i] <= list[i + 1]) {
            list[i + 1] = i * 2;
        }
    }
}

public static void main(String[] args) {
    int a = 4;
    int b = 8;
    int[] data = {5, 10, 15};
    a = mystery(b, data); // 108 [5, 10, 14]
    System.out.println(a + " " + b + " " + Arrays.toString(data)); // 20 8 [5, 10, 14]
    System.out.println();

    int[] x = {1, 2};
    int[] y = {3, 4, 5};
    mystery(x, y); // [3,4,5] [1,2]
    System.out.println(Arrays.toString(x) + " " + Arrays.toString(y)); // [1,2] [3,4,5]
    System.out.println();

    int[] data1 = {4, 1, 3};
    int[] data2 = {2, 1, 3, 2};
    int[] data3 = {1, 1, 1, 1, 8};
    mystery(data1);
    System.out.println("data1 = " + Arrays.toString(data1)); // data1=[4,1,2]
    mystery(data2);
    System.out.println("data2 = " + Arrays.toString(data2)); // data2=[2,1,2,4]
    mystery(data3);
    System.out.println("data3 = " + Arrays.toString(data3)); // data3=[1,1,2,1,6]
    System.out.println();
}
```

108 [5, 10, 14]
20 8 [5, 10, 14]

[3, 4, 5] [1,2] // 자바는 pass-by-value 방식이라서, 내부에서는 바뀌지만
[1, 2] [3, 4, 5] // 자바는 pass-by-value 방식이라서, 외부에는 영향을 주지 않는다.

data1 = [4, 1, 2]
data2 = [2, 1, 2, 4]
data3 = [1, 1, 2, 1, 6]

학과 _____ 학번 _____ 이름 _____

3. 다음은 클래스 상속관계에서 다형성을 보여주고 있다. 아래의 질문에 답하라. (30점)

```
interface I {
    void method1(int v);
}
public abstract class A implements I {
    public abstract void method2();
    public void method3() {
        System.out.println(this);
    }
    public String toString() {
        return "a";
    }
}
public class D extends A {
    public void method1(int v) {
        System.out.println("d1 " + v);
    }
    public void method2() {
        System.out.println("d2");
    }
    public String toString() {
        return "d";
    }
}
public class C extends A {
    protected int m = 10;
    public void method1(int v) {
        System.out.println("c1 " + v);
    }
    public void method2() {
        System.out.println("c2 " + m);
    }
    public String toString() {
        return "c";
    }
}
public class B extends C {
    protected int m = 20;
    public void method2() {
        System.out.println("b2 " + this.m + " " + super.m); // (3.3)
    }
    public String toString() {
        return "b";
    }
}
public class E extends C {
    public void method2() {
        System.out.println("e2 " + this.m + " " + super.m); // (3.3)
    }
}
```

3.1 interface와 abstract class가 무엇인지 간단히 설명하라. (5점)

인터페이스는 클래스에서 구현해야 하는 동작을 지정하는데 사용하는 추상형. 자바에서 인터페이스는 다중 상속이 가능하다.

추상 클래스는 여러 기존의 클래스에서 공통된 부분을 추상화한 것. 추상 클래스를 상속하는 클래스에게 그 구현을 강제화하는 클래스. 하나 이상의 추상 메소드를 포함하는 클래스이다.

학과 _____ 학번 _____ 이름 _____

3.2 abstract method가 무엇인지 간단히 설명하라. 위의 코드에서 예를 찾아서 보여라. (5점)

추상 메소드는 메소드만 선언해놓고 구현 내용은 없는 것으로, 추상 메소드를 상속받은 클래스는 반드시 추상메소드를 override해서 구현해야 한다.

위의 코드에서는 method1과 method2가 추상메소드이다.

3.3 클래스 B와 E의 method2()에 this.m과 super.m의 값을 말하고 자세히 설명하라. (5점)

클래스 B의 method2()에서 사용되는 this.m은 클래스 B에서 다시 정의한 멤버 m으로 20, super.m은 클래스 C의 멤버 m으로 10 이다. 따라서 b2 20 10

클래스 E의 method2()에서 사용되는 this.m과 super.m은 클래스 C의 멤버 m이므로 모두 10 이다. 따라서 e2 10 10

3.4 다음 코드의 실행결과를 적고, 코드에 upcasting과 downcasting을 표시하라. (5점)

```
A a = new B(); // upcasting
System.out.println(a); // dynamic binding A.toString() => B.toString() “b”
B b = new B();
Object o = b; // upcasting
((B)o).method2(); // downcasting B.method2() “b2 20 10”
System.out.println((B)o); // downcasting, B.toString() “b”
System.out.println((C)o); // downcasting, C.toString() => B.toString() “b”
System.out.println(o); // Object.toString() => B.toString() “b”
System.out.println();
```

```
b // B.toString()
b2 20 10 // B.method2()
b // B.toString()
b // C.toString() => B.toString()
b // Object.toString() => B.toString()
```

3.5 다음 코드의 실행결과를 적고 이유를 자세히 설명하라 (dynamic binding에 주의). (10점)

```
A[] elements = {new C(), new D(), new E()};
for (int i = 0; i < elements.length; i++) {
    System.out.println(elements[i]);
    elements[i].method1(i);
    elements[i].method2();
    elements[i].method3();
}
```

```
c // A.toString() => C.toString()
c1 0 // A.method1(0) => C.method1(0)
c2 10 // A.method2() => C.method2()
c // A.method3() 호출하는데 그 내부의 this는 c
d // A.toString() => D.toString()
d1 1 // A.method1(1) => D.method1(1)
d2 // A.method2() => D.method2()
d // A.method3() 호출하는데 그 내부의 this는 d
c // A.toString() => E.toString() 없으므로 C.toString() 호출
c1 2 // A.method1(2) => E.method1(2) 없으므로 C.method1(2) 호출
e2 10 10 // A.method2() => E.method2()
c // A.method3() 호출하는데 그 내부의 this는 e, 하지만 E.toString()이 없으므로 C.toString() 호출
```

학과 _____ 학번 _____ 이름 _____

4. 다음은 Value와 TripleValue 클래스이다. Collection 관련 질문에 답하라. (30점)

```
import java.util.*;
public class Value implements Comparable<Value> {
    private int x;
    private int y;
    public Value() {
        this(0, 0);
    }
    public Value(int x, int y) {
        set(x, y);
    }
    public void set(int x, int y) {
        this.x = x;
        this.y = y;
    }
    public int getX() {
        return x;
    }
    public int getY() {
        return y;
    }
    @Override
    public String toString() {
        return x + "," + y;
    }
    @Override
    public boolean equals(Object other) {
        if (this == other) return true;
        if (other instanceof Value) {
            Value that = (Value) other;
            return x == that.x && y == that.y;
        }
        return false;
    }
    @Override
    public int hashCode() {
        return Objects.hash(x, y);
    }
    @Override
    public int compareTo(Value other) {
        return x - other.x;
    }
    public static Comparator<Value> YComparator = new Comparator<Value>() {
        public int compare(Value v1, Value v2) {
            return v1.y - v2.y;
        }
    };
}
import java.util.*;
public class TripleValue extends Value {
    private int z;
    public TripleValue() {
        this(0, 0, 0);
    }
    public TripleValue(int x, int y, int z) {
        super(x, y);
        this.z = z;
    }
    public void set(int x, int y, int z) {
        super.set(x, y);
```

학과 _____ 학번 _____ 이름 _____

```
        this.z = z;
    }
    public int getZ() {
        return z;
    }
    @Override
    public String toString() {
        return getX() + "," + getY() + "," + z;
    }
    @Override
    public boolean equals(Object other) {
        if (this == other) return true;
        if (other instanceof TripleValue) {
            TripleValue that = (TripleValue)other;
            return getX() == that.getX() && getY() == that.getY() && z == that.z;
        }
        return false;
    }
    @Override
    public int hashCode() {
        return Objects.hash(getX(), getY(), z);
    }
    public static Comparator<TripleValue> ZComparator = new Comparator<TripleValue>() {
        public int compare(TripleValue v1, TripleValue v2) {
            return v1.z - v2.z;
        }
    };
}
```

4.1 다음 코드의 실행결과와 같도록, values를 사용하여 List<Value> list를 작성하라. (5점)

```
        Value[] values = {
                new Value(1, 2),
                new Value(1, 2),
                new TripleValue(-3, -4, -5),
                new TripleValue(-3, -4, -5),
                new TripleValue(7, 8, 9)
        };
        for (Value value : values) System.out.println(value);
```

```
        List<Value> list = new ArrayList<Value>();
        for (Value value : values) {
                list.add(value);
        }
        for (Value value : list) {
                System.out.println(value);
        }
```

4.2 위 4.1의 list를 사용하여, 각각 X, Y 값으로 sort하는 코드를 작성하라. (5점)

```
list.sort(null); // 기본 X값으로 sort. Comparable<Value>의 compareTo를 사용
list.sort(Value.YComparator); // Y값으로 sort는 Comparator<Value>의 compare 사용
//만약 list의 element가 TripleValue였다면 list.sort(TripleValue.ZComparator) 사용 가능
// 또는
Collections.sort(list); // 기본 X값으로 sort. Comparable<Value>의 compareTo를 사용
Collections.sort(list, Value.YComparator); // Y값으로 sort는 Comparator<Value>의 compare 사용
```

학과 _____ 학번 _____ 이름 _____

4.3 위 4.1의 values를 사용한 HashMap 코드이다. 다음 실행 결과를 적어라. (5점)

```
Map<Value, Integer> map = new HashMap<Value, Integer>();
for (int i = 0; i < values.length; i++) {
    map.put(values[i], i+1);
}
for (Map.Entry<Value, Integer> e : map.entrySet()) {
    System.out.println(e.getKey() + " => " + e.getValue());
}
map.remove(values[2]);
for (Map.Entry<Value, Integer> e : map.entrySet()) {
    System.out.println(e.getKey() + " => " + e.getValue());
}
```

```
1, 2 => 2
7, 8, 9 => 5
-3, -4, -5 => 4
// remove후
1, 2 => 2
7, 8, 9 => 5
```

4.4 위의 4.3 실행결과와 같도록, foreach 대신 iterator와 while를 사용하여 동일한 코드를 작성하라. (5점)

```
Iterator<Value> it = map.keySet().iterator();
while (it.hasNext()) {
    Value key = it.next();
    Integer value = map.get(key);
    System.out.println(key + " => " + value);
}
```

4.5 다음 코드의 실행결과를 적고 ==과 equals와 hashCode를 자세히 설명하라. (10점)

```
TripleValue v1 = new TripleValue(1,2,3);
TripleValue v2 = new TripleValue(1,2,3);
TripleValue v3 = v1;
System.out.println("v1=" + v1);
System.out.println("v2=" + v2);
System.out.println("v3=" + v3);
System.out.println("v1 == v2 " + (v1 == v2));
System.out.println("v1 == v3 " + (v1 == v3));
System.out.println("v1 equals v2 " + v1.equals(v2));
System.out.println("v1 equals v3 " + v1.equals(v3));
System.out.println("v1.hashCode == v2.hashCode " + (v1.hashCode() == v2.hashCode()));
System.out.println("v1.hashCode == v3.hashCode " + (v1.hashCode() == v3.hashCode()));
```

```
v1=1,2,3
v2=1,2,3
v3=1,2,3
v1 == v2 false
v1 == v3 true // 레퍼런스가 같은 v1 == v3
v1 equals v2 true // 객체의 내용이 같으므로 (equals 재정의 되어 있으므로)
v1 equals v3 true // 객체의 내용이 같으므로 (equals 재정의 되어 있으므로)
v1.hashCode == v2.hashCode true // hashCode 재정의되어 있으므로
v1.hashCode == v3.hashCode true // hashCode 재정의되어 있으므로
== 연산자는 reference type의 경우, reference 가 같은지를 비교한다.
equals 메소드는 두 객체의 내용이 같은지를 비교한다. hashCode 메소드는 객체의 내용으로 hashCode 값을 생성한다. 따라서 객체의 내용이 같을 경우 같은 hashCode를 갖게 된다.
hashCode가 override되어 있을 경우, HashMap이나 HashSet을 사용하는 collection에서 동일한 Key로 사용할 수 없다.
```

학과 _____ 학번 _____ 이름 _____

5. 다음은 TripleCalculatorFrame 클래스이다. 아래 빈 칸에 코드를 채워라. (20점)

```
// TripleOperator
public enum TripleOperator {
    MIN, MAX, MEDIAN, MEAN, SUM;
    public static TripleOperator nameOf(int value) {
        switch (value) {
        case 0:  return MIN;
        case 1:  return MAX;
        case 2:  return MEDIAN;
        case 3:  return MEAN;
        case 4:  return SUM;
        default: return null;
        }
    }
    public double calculate(int x, int y, int z) {
        switch (this) {
        case MIN:       return Math.min(Math.min(x, y), z);
        case MAX:       return Math.max(Math.max(x, y), z);
        case MEDIAN:    return Math.max(Math.min(x,y), Math.min(Math.max(x,y),z));
        case MEAN:      return (x + y + z) / 3.0;
        case SUM:       return x + y + z;
        default:        throw new AssertionError("Unknown operations " + this);
        }
    }
}
// TripleCalculator
public class TripleCalculator {
    private TripleValue value; // x,y,z value
    private TripleOperator op; // operator
    public TripleCalculator() {
        set(0, 0, 0, TripleOperator.MIN);
    }
    public void set(int x, int y, int z, TripleOperator op) {
        this.value = new TripleValue(x, y, z);
        this.op = op;
    }
    public double getW() {
        return op.calculate(value.getX(), value.getY(), value.getZ());
    }
}
// TripleCalculatorFrame
import java.awt.event.*;
import javax.swing.*;
public class TripleCalculatorFrame extends JFrame implements ActionListener, KeyListener {
    JLabel[] labels = new JLabel[4];
    JTextField[] textfields = new JTextField[4];
    JRadioButton[] rbuttons = new JRadioButton[5];
    String[] names = {"MIN", "MAX", "MEDIAN", "MEAN", "SUM"};
    ButtonGroup rgroup = new ButtonGroup();
    JPanel rpanel = new JPanel();
    TripleCalculator calc = new TripleCalculator();
    public TripleCalculatorFrame() {
        super("TripleCalculatorFrame");
        this.setLayout(null);
        for (int i = 0; i < 3; i++) {
            labels[i] = new JLabel("숫자 " + (i+1));
            labels[i].setBounds(10, 10 + 30 * i, 50, 25);
            textfields[i] = new JTextField("0", 10);
            textfields[i].setBounds(100, 10 + 30 * i, 200, 25);
            textfields[i].addKeyListener(this); // event
```

학과 _____ 학번 _____ 이름 _____

```
            this.add(labels[i]);
            this.add(textfields[i]);
        }
        for (int i = 0; i < 5; i++) {
            rbuttons[i] = new JRadioButton(names[i]);
            rbuttons[i].addActionListener(this); // event
            rgroup.add(rbuttons[i]);
            rpanel.add(rbuttons[i]);
        }
        rbuttons[0].setSelected(true);
        rpanel.setBounds(10, 100, 300, 25);
        this.add(rpanel);
        labels[3] = new JLabel("결과");
        labels[3].setBounds(10, 150, 50, 25);
        textfields[3] = new JTextField(10);
        textfields[3].setBounds(100, 150, 200, 25);
        this.add(labels[3]);
        this.add(textfields[3]);
        setSize(350, 250);
        setResizable(false);
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    private int selectOperator() {
        for (int i = 0; i < 5; i++) {
            if (rbuttons[i].isSelected()) {
                return i;
            }
        }
        return 0;
    }
    private void calculate(int index) {
        int x = Integer.parseInt(textfields[0].getText());
        int y = Integer.parseInt(textfields[1].getText());
        int z = Integer.parseInt(textfields[2].getText());
        TripleOperator op = TripleOperator.nameOf(index);
        calc.set(x, y, z, op);
        labels[3].setText(names[index]);
        textfields[3].setText("" + calc.getW());
    }
    public void actionPerformed(ActionEvent e) {
        JRadioButton rbutton = (JRadioButton) e.getSource();
        for (int i = 0; i < 5; i++) {
            if (rbutton == rbuttons[i]) {
                calculate(i); // 계산
            }
        }
    }
    public void keyTyped(KeyEvent e) {}
    public void keyReleased(KeyEvent e) {}
    public void keyPressed(KeyEvent e) {
        int key = e.getKeyCode();
        if (key == KeyEvent.VK_ENTER) {
            calculate(selectOperator()); // 계산
        }
    }
}
```

-끝-