オブジェクト指向言語・期末テスト問題用紙
（2021年 07月 29日・ 08:50 ～ 10:20）

解答上、その他の注意事項

1. 問題は、問 Ⅰ～Ⅶ まである。うち、<u>問 Ⅰ～Ⅲ は中間テストの代替問題である。</u>
    - 中間テストの得点が 6 割に達しなかったものは、 代替問題を解答すれば、6 割を上限として良いほうの点数を採用する。
    - 正当な事情があって中間テストを欠席した者も代替問題を解答すること。（この場合は、上限は設けない。）
2. 解答用紙の右上の欄に学籍番号・名前を記入すること。
3. 解答欄を間違えないよう注意すること。
4. 解答中の文字 (特に a と d)がはっきりと区別できるよう注意すること。
5. 持ち込みは不可である。筆記用具・時計・学生証以外のものは、かばんの中などにしまうこと。
6. テストの配点は 70 点（うち中間テストの代替問題の問 Ⅰ～Ⅲの配点は 40 点）である。合格はレポート（創造工学部生は「オブジェクト指向言語演習」の課題）の得点を加点して、 100 点満点中 60 点以上とする。

すべての問に対する補足

プログラムの空欄を埋める問題では、解答が長くなる可能性があるので、下の省略形（◯囲み文字）を用いても良い。（必ず ◯ で囲むこと。）

Ⓐ `ActionListener` ⓐA `addActionListener` ⒶE `ActionEvent` Ⓚ `KeyListener`
ⓐK `addKeyListener` ⓀE `KeyEvent` Ⓜ `MouseListener` ⓐM `addMouseListener`
ⓂE `MouseEvent` ⓟl `System.out.println` ⓟf `System.out.printf`

また、参考のために問題用紙の末尾に授業配布プリントの LeftRightButton.java, LeftRightButton2.java, Guruguru.java, Point.java, ColorPoint.java, Quadratic.kt, SequenceTest.kt のソースを掲載する。（GUI アプリケーションは `main` メソッドは省略している。）

I. 次の (ⅰ)～(ⅱ) の Java に関する多肢選択問題に答えよ。解答は各問の指示する選択肢から選べ。

(ⅰ) 変数 str に "135" という文字列（String 型）が代入されているとき、 135 という整数（int 型）を返す式は次のうち、どれか? 一つ選べ。

(A). atoi(str)　(B). Integer.toString(str)
(C). Integer.parseInt(str)　(D). String.valueOf(str)

(ⅱ) 変数 list を、要素の型が String であるような、空の ArrayList に初期化するための宣言として正しいものはどれか? 一つ選べ。

(A). ArrayList<String> list = ArrayList<String>()
(B). ArrayList<String> list = new ArrayList<>()
(C). String<ArrayList> list = new String<>()
(D). new String<ArrayList> list = String<ArrayList>()

II. 次の枠内の文章は java.util.Arrays クラスの copyOf メソッド、 java.lang.String クラスの contains メソッド、join メソッド、split メソッド、の Java™ API 仕様からの抜粋である。（ただし簡単にするため、String 型（の配列）に特化した形に書き換え、 問題を解くのに関係ない部分は割愛している。）

java.util

クラス Arrays

copyOf

```
public static String[] copyOf(String[] original, int newLength)
```

指定された配列をコピーし、そのコピーが指定された長さになるように、必要に応じて切り詰めるかナルでパディングします。 （以降略）

パラメーター:
　original - コピーされる配列
　newLength - 返されるコピーの長さ
戻り値:
　指定された長さにするために切り詰められた、またはnullでパディングされた元の配列のコピー
例外:
　（略）

java.lang

クラス String

contains

```
public boolean contains(String s)
```

この文字列が指定された文字列を含む場合に限りtrueを返します。

パラメーター:
　s - 検索する文字列
戻り値:

この文字列がsを含む場合はtrue。そうでない場合はfalse

join

```
public static String join(String delimiter, String[] elements)
```

指定されたdelimiterのコピーを使用して結合されたString配列の要素のコピーからなる新しいStringを返します。

次に例を示します。

```
String message = String.join("-", new String[]{"Java", "is", "cool"});
// message returned is: "Java-is-cool"
```

パラメーター:
delimiter - 各要素を区切る区切り文字
elements - 結合する要素の配列。
戻り値:
delimiterで区切られたelementsからなる新しいString
例外:
（略）

split

```
public String[] split(String regex)
```

この文字列を、指定された正規表現に一致する位置で分割します。

たとえば、次の式が指定された場合の、文字列"boo:and:foo"の結果を示します。

正規表現	結果
:	{ "boo", "and", "foo" }
o	{ "b", "", ":and:f" }

パラメーター:
regex - 正規表現の区切り
戻り値:
この文字列を指定された正規表現に一致する位置で分割して計算された文字列の配列

次の StringSample クラスは、これらのメソッドを使って、文字列を空白で単語に分解し、分解してできた配列のコピーをとり、一つの配列は単語の中に “ion” を含むものの後ろに !! をつけ、もう一つの配列は単語のなかに “er” を含むものの後ろに ?? をつけ、最後に結合して、出力するプログラムである。 以下の問に答えよ。

ファイル名 StringSample.java

```
1  import java.util.Arrays;
2
3  public class StringSample {
4      public static void main(String[] args) {
5          String str = "classes and interfaces for cryptographic operations";
6          String[] words1 = ( i );
7          String[] words2 = ( ii );
8
9          for (int i = 0 ; i < words1.length; i++) {
10             String w = words1[i];
11             if ( (iii) ) {
12                 words1[i] = w + "!!";
13             }
14         }
```

```
15
16          for (int j = 0; j < words2.length; j++) {
17              String w = words2[j];
18              if (          (iv)          ) {
19                  words2[j] = w + "??";
20              }
21          }
22
23          System.out.println(                    (v)                    );
24          System.out.println(                    (vi)                   );
25      }
26  }
27
```

このプログラムは次のように出力する。以下の問に答えよ。

```
classes and interfaces for cryptographic operations!!
classes and interfaces?? for cryptographic operations??
```

(i) には、文字列 str を空白 " " 区切りで単語に分割する式が入る。この式を答えよ

(ii) には、配列 words1 のコピーを返す式が入る。この式を答えよ。

(iii), (iv) には、文字列 w が、それぞれ文字列 "ion", "er" を含んでいるかを表す式が入る。この式を答えよ。

(v), (vi) には、それぞれ、words1, words2 の各要素を結合した文字列を返す式が入る。この式を答えよ。

III. 次の GraphSample クラスはとある数式のグラフを表示し、「縮小」ボタンをクリックすればグラフを縮小、「拡大」ボタンをクリックすればグラフを拡大表示する GUI アプリケーションである。GraphSample は JPanel を継承している。

ファイル名 GraphSample.java

```
 1  import javax.swing.*;
 2  import java.awt.*;
 3  import java.awt.event.*;
 4  import static java.lang.Math.sin;
 5
 6  public class GraphSample                    (i)                    {
 7      private int scale = 0;
 8      private final JButton minus, plus;
 9      private final int unit = 16, width = 80, height = 80, margin = 20;
10
11      public GraphSample() {
12          setPreferredSize(new Dimension(width * 2, margin + height * 2));
13          minus = new JButton("縮小");
14          plus  = new JButton("拡大");
15          minus.addActionListener(this);
16          plus.addActionListener(this);
17          add(minus); add(plus);
18      }
19
20      private static double f(double x) {
21          return sin(x) + sin(x / 3) + sin(x / 5);
22      }
23
24      @Override
25      protected void paintComponent(Graphics g) {
26          super.paintComponent(g);
27
28          double ex = unit * Math.pow(2, scale);
29          double xmax = width / ex;
30          double dx = 2 * xmax;
31          int n = 64;
32
```

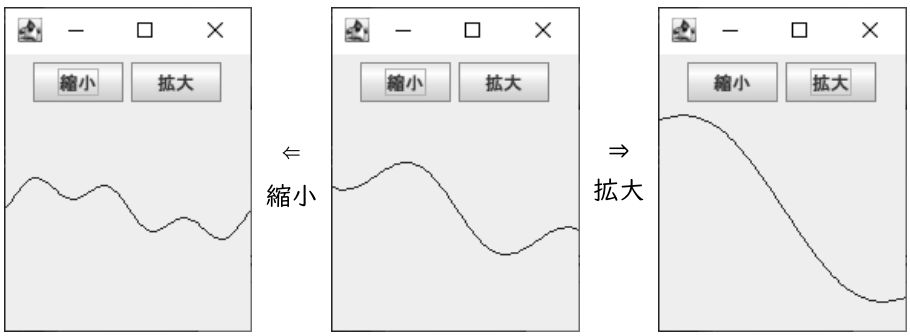
```
        for (int i = 0; i < n; i++) {
            double x0 = - xmax + dx * i / n;
            double y0 = f(x0);
            double x1 = - xmax + dx * (i + 1) / n;
            double y1 = f(x1);
            g.setColor(Color.BLUE);
            g.drawLine(width + (int)(x0 * ex), margin + height + (int)(y0 * ex),
                       width + (int)(x1 * ex), margin + height + (int)(y1 * ex));
        }
    }

    public void actionPerformed(ActionEvent e) {
        Object source = e.getSource();
        if (source == minus) {
            scale--;
        } else {
            scale++;
        }
        repaint();
    }

    /* main は省略する */
}

```

実行例:



上の (i) の空欄を埋めよ。

このプログラムを内部クラスを用いて次のように書き換えるとき、(ii) ～ (v) の空欄を埋めよ。

```
// import は変わらないので省略する。

public class GraphSample2 [          ( ii )          ] {
    private int scale = 0;
    private final int unit = 16, width = 80, height = 80, margin = 20;

    private class ButtonListener [          ( iii )          ] {
        private int ds;
        public ButtonListener(int d) {
            ds = d;
        }
        public void actionPerformed(ActionEvent e) {
            scale += ds;
            repaint();
        }
    }

    public GraphSample2() {
        setPreferredSize(new Dimension(width * 2, margin + height * 2));
        JButton minus = new JButton("縮小");
        JButton plus  = new JButton("拡大");
        minus.addActionListener([          ( iv )          ]);
        plus.addActionListener([          ( v )          ]);
        add(minus); add(plus);
```

```
    }
    // f, paintComponent は変わらないので省略する。
    // main は省略する。
}

```

IV. 次の (i)〜(ii) の Kotlin プログラムはどのように出力するか？答えよ。なお、take(n) メソッドは先頭の *n* 個の要素からなる有限列を取り出し、 toList() メソッドは、出力するためにシーケンスをリストに変換する。

(i)

```
fun foo(n: Int): Pair<Int, Int> {
    if (n == 0) return Pair(0, 1)
    val (m, n) = foo(n - 1)
    return Pair(n, m + n)
}

fun main() {
    val (a, b) = foo(4)
    println(b)
}

```

(ii)

```
fun bar(m: Int) = sequence {
    var n = m
    while (true) {
        yield(n)
        if (n % 2 == 0) {
            n /= 2
        } else {
            n = 3 * n + 1
        }
    }
}

fun main() {
    println(bar(3).take(5).toList())
}

```

V. 次の (i)〜(ii) の多肢選択問題に答えよ。

(i) 次の選択肢の中でオブジェクト指向型に分類される言語はどれか？もっともふさわしいものを一つ選べ。

(A). C (B). Fortran (C). Smalltalk (D). Prolog (E). Haskell

(ii) C, Java, Algol, Kotlin の 4 つの言語を登場順（古い順）に並べ替えるとき、正しいものを一つ選べ。

(A). C, Java, Algol, Kotlin
(B). Algol, C, Java, Kotlin
(C). C, Algol, Kotlin, Java
(D). C, Kotlin, Algol, Java
(E). Algol, Java, C, Kotlin
(F). Java, C, Kotlin, Algol

VI. ゲーム用のクラス階層を設計するために、テスト用のいくつかのクラスを作成する。次に定義されるクラス Monster を継承して、

ファイル名 Monster.java

```
class Monster {
    public String name;
    protected int fullHp;
    protected int hp;
    protected int combat;
    public Monster(String n, int h, int c) {
        name   = n;
        fullHp = hp = h;
        combat = c;
    }
    public int calcÐamage(int attack) {
        int hp0 = hp;
        hp -= attack;
        if (hp < 0) hp = 0;
        return hp0 - hp;
    }
    public int calcCombat() {
        if (hp > 0) return combat;
        return 0;
    }
}

```

3つのクラス Ðavid, Ellen, Frank を定義する。

ファイル名 David.java

```
public class Ðavid extends Monster {
  private int defense;
  public Ðavid() {
      super("Ðavid", 9, 5);
      defense = 35;
  }
  @Override
  public int calcÐamage(int attack) {
      int result = attack - defense;
      if (result <= 0) result = 1;
      return super.calcÐamage(result);
  }
}

```

ファイル名 Ellen.java

```
public class Ellen extends Monster {
  public Ellen() {
      super("Ellen", 50, 7);
  }
  @Override
  public int calcÐamage(int attack) {
      if (2 * hp >= fullHp) { // HP が 50% 以上なら
          int hp0 = hp;
          super.calcÐamage(attack);
          if (hp <= 0) hp = 1;  // 耐える
          return hp0 - hp;
      }
      return super.calcÐamage(attack);
  }
  @Override
  public int calcCombat() {
      if (hp * 10 < fullHp) { // HP が 10% 未満なら
          return 10 * super.calcCombat(); // 攻撃力 UP
      }
      return super.calcCombat();
  }
}

```

ファイル名 Frank.java

```
public class Frank extends Monster {
  public Frank() {
      super("Frank", 80, 8);
  }
  @Override
  public int calcCombat() {
      if (hp * 2 >= fullHp) { // HP が 50% 以上なら
          return 3 * super.calcCombat(); // 攻撃力 UP
      }
      return super.calcCombat();
  }
}
```

ファイル名 MosterTest.java

```
public class MonsterTest {
    public static void main(String[] args) {
        int[] attacks = {20, 30, 40};
        Monster[] characters = {
            new David(), new Ellen(), new Frank()
        };

        for (int a: attacks) {
            for (Monster ch: characters) {
                int d = ch.calcDamage(a);
                int c = ch.calcCombat();
                System.out.printf(
                    "%s に %2d の攻撃、%2d のダメージ、%s から %2d の攻撃%n",
                    ch.name, a, d, ch.name, c);
            }
        }
    }
}
```

(ⅰ). Monster の 4 つのフィールド name, fullHp, hp, combat は、 public や protected と宣言されているが、これを private に変更しても（ここに掲載された 5 つのクラスをコンパイル・実行するとき）エラーにならないフィールドはどれか？

(A). name (B). fullHp (C). hp (D). combat

(ⅱ). MonsterTest クラスの main メソッドを実行するとき、出力はどうなるか？

なお、解答用紙に記入するとき、空白の有無や数は気にしなくて良い。

VII. 次の CircularText は文字列を円形にしてアニメーションにする GUI アプリケーションである。CircularText クラスは JPanel を継承している。

ファイル名 CircularText.java

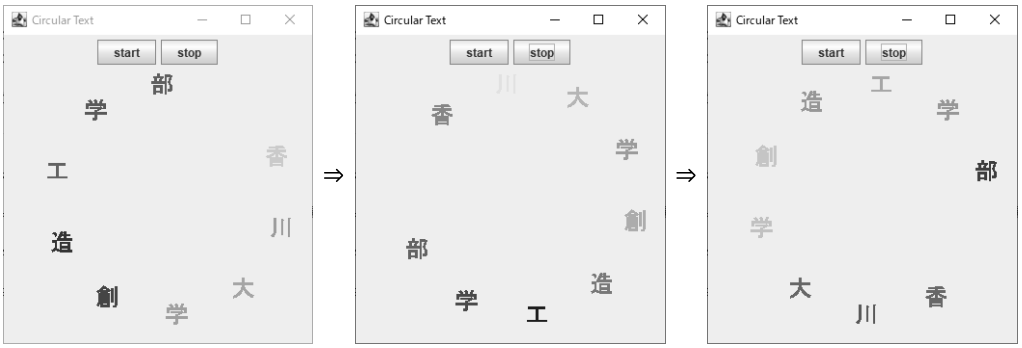
```
import java.awt.*;
import javax.swing.*;

public class CircularText [                    (ⅰ)                    ] {
    volatile Thread moi = null;
    volatile int step = 0;

    public CircularText() {
        setPreferredSize(new Dimension(320, 320));
        JButton startBtn = new JButton("start");
        startBtn.addActionListener(e -> startThread());
        JButton stopBtn = new JButton("stop");
        stopBtn.addActionListener(e -> stopThread());
        setLayout(new FlowLayout());
        add(startBtn); add(stopBtn);
```

```
16          startThread();
17      }
18
19      public void startThread() {
20          if (moi == null) {
21              moi = [          ( ii )          ];
22              moi.start();
23          }
24      }
25
26      public void stopThread() {
27          moi = null;
28      }
29
30      @Override
31      protected void paintComponent(Graphics g) {
32          super.paintComponent(g);
33          final String msg = "香川大学創造工学部  ";
34          final int len = msg.length();
35          final Font f = new Font(Font.MONOSPACED, Font.BOLD, 24);
36          g.setFont(f);
37          for (int i = 0; i < len; i++) {
38              double theta = Math.PI + 2 * Math.PI / len * (i - step / 10.0);
39              double x = 160 + 120 * Math.cos(theta);
40              double y = 180 + 120 * Math.sin(theta);
41              float h = 1.0f / len * (i - step / 20f);
42              g.setColor(Color.getHSBColor(h, 1, 1));
43              g.drawString(msg.substring(i, i + 1), (int)x, (int)y);
44          }
45      }
46
47      public void run() {
48          Thread me = Thread.currentThread();
49          while ([          ( iii )          ]) {
50              repaint();
51              try {
52                  Thread.sleep(30);
53              } catch (InterruptedException e) {}
54              step++;
55          }
56      }
57      /* main は省略する */
58  }
59
```

実行例:



空欄 (i) ～ (iii) を埋めてプログラムを完成せよ。（空欄 (ii), (iii) は文法上、式である。）

以下に参考のために授業配布プリントの LeftRightButton.java, LeftRightButton2.java, Guruguru.java, Point.java, ColorPoint.java, Quadratic.kt, SequeceTest.kt のソースを掲載する。

ファイル LeftRightButton.java

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

public class LeftRightButton extends JPanel implements ActionListener {
    private int x = 20;
    private JButton lBtn, rBtn;

    public LeftRightButton() {
        setPreferredSize(new Dimension(200, 70));
        lBtn = new JButton("Left");
        rBtn = new JButton("Right");
        lBtn.addActionListener(this);
        rBtn.addActionListener(this);
        setLayout(new FlowLayout());
        add(lBtn); add(rBtn);
    }

    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawString("HELLO WORLD!", x, 55);
    }

    public void actionPerformed(ActionEvent e) {
        Object source = e.getSource();
        if (source == lBtn) {      // lBtnが押された
            x -= 10;
        }
        else if (source == rBtn) { // rBtnが押された
            x += 10;
        }
        repaint();
    }

    public static void main(String[] args) { /* 省略 */ }
}

```

ファイル LeftRightButton2.java

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

public class LeftRightButton2 extends JPanel {
    private int x = 20;

    public LeftRightButton2() {
        setPreferredSize(new Dimension(200, 70));
        JButton lBtn = new JButton("Left");
        JButton rBtn = new JButton("Right");
        lBtn.addActionListener(new LeftListener());
        rBtn.addActionListener(new RightListener());
        setLayout(new FlowLayout());
        add(lBtn); add(rBtn);
    }

    private class LeftListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            x -= 10;
            repaint();
        }
    }

    private class RightListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            x += 10;
            repaint();
        }
    }

```

```
    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawString("HELLO WORLD!", x, 55);
    }

    public static void main(String[] args) { /* 省略 */ }
}

```

ファイル Guruguru.java

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

public class Guruguru extends JPanel implements Runnable {
    private int r = 50;
    private volatile int x = 110, y = 70 ;
    private double theta = 0; // 角度
    private volatile Thread thread = null;

    public Guruguru() {
        setPreferredSize(new Dimension(200, 180));
        JButton startBtn = new JButton("start");
        startBtn.addActionListener(e -> startThread());
        JButton stopBtn = new JButton("stop");
        stopBtn.addActionListener(e -> stopThread());
        setLayout(new FlowLayout());
        add(startBtn); add(stopBtn);
        startThread();
    }

    private void startThread() {
        if (thread == null) {
            thread = new Thread(this);
            thread.start();
        }
    }

    private void stopThread() {
        thread = null;
    }

    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g); // スーパークラスの paintComponent を呼び出す
        // 全体を背景色で塗りつぶす。
        g.drawString("Hello, World!", x, y);
    }

    public void run() {
        Thread thisThread = Thread.currentThread();
        for (; thread == thisThread; theta += 0.02) {
            x = 60 + (int)(r * Math.cos(theta)); y = 100 - (int)(r * Math.sin(theta));
            repaint();  // paintComponent を間接的に呼出す
            try {
                Thread.sleep(30); // 30 ミリ秒お休み
            } catch (InterruptedException e) {}
        }
    }

    public static void main(String[] args) { /* 省略 */ }
}

```

ファイル Point.java

```
public class Point {
  // フィールド(メンバー変数)
  public int x;
  public int y;

  // メソッド（メンバー関数）
  public void move(int dx, int dy) {
    x += dx;
    y += dy;
  }

  public double distance() {
```

```
    return Math.sqrt(x * x + y * y);
  }

  public void print() {
    System.out.printf("(%d, %d)", x, y);
  }

  public void moveAndPrint(int dx, int dy) {
    print(); move(dx, dy); print();
  }

  // コンストラクター
  public Point(int x0, int y0) {
    x = x0; y = y0;
  }
}

```

ファイル ColorPoint.java

```
public class ColorPoint extends Point {
    public String[] cs = {
        "black", "red", "green", "yellow",
        "blue", "magenta", "cyan", "white" };
    public String color;

    @Override
    public void print() {
        System.out.printf("<font color='%s'>", getColor()); // 色の指定
        System.out.printf("(%d, %d)", x, y); // super.print(); でも可
        System.out.print("</font>");                        // 色を戻す
    }

    public void setColor(String c) {
        int i;
        for (i = 0; i < cs.length; i++) {
            if (c.equals(cs[i])) {
                color = c; return;
            }
        }
        // 対応する色がなかったら何もしない。
    }

    public ColorPoint(int x, int y, String c) {
        super(x, y);
        setColor(c);
        if (color == null) color = "black";
    }

    public String getColor() {
        return color;
    }
}

```

ファイル Quadratic.kt

```
import kotlin.math.*

fun quadratic(a: Double, b: Double, c: Double): Pair<Double, Double> {
    val d = b * b - 4 * a * c
    val sq = sqrt(d)
    return Pair((-b + sq) / (2 * a), (-b - sq) / (2 * a))
}

fun main() {
    val a = 1.0; val b = -1.0; val c = -1.0
    val (x1, x2) = quadratic(a, b, c)
    println("方程式の解は、$x1、$x2 です。")
    // 出力 ⇒ 方程式の解は、1.618033988749895、-0.6180339887498949 です。
}
```

ファイル SequenceTest.kt

```
fun main() {
    val nums = generateSequence(1) { x -> x + 3 }
    println(nums.take(10).toList())
    // 出力 ⇒ [1, 4, 7, 10, 13, 16, 19, 22, 25, 28]
```

```
    val fibs = sequence {
        var a = 1; var b = 1
        while (true) {
            yield(a)
            val c = a
            a = b; b += c
        }
    }
    println(fibs.take(10).toList())
    // 出力 ⇒ [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
}
```

（計算用紙）

オブジェクト指向言語・期末テスト解答用紙（**2021** 年 **07** 月 **29** 日）

学籍番号		氏名	

I. (4 × 2)

(ⅰ)		(ⅱ)	

II. (4, 4, 2, 2, 2, 2)

(ⅰ)	
(ⅱ)	
(ⅲ)	
(ⅳ)	
(ⅴ)	
(ⅵ)	

III. (4, 4, 4, 2, 2)

(ⅰ)	
(ⅱ)	
(ⅲ)	
(ⅳ)	
(ⅴ)	

IV. (3 × 2)

(ⅰ)	
(ⅱ)	

V. (3 × 2)

(ⅰ)		(ⅱ)	

VI. (3, 6)

(ⅰ)	
(ⅱ)	に　の攻撃、　のダメージ、　から　の攻撃 〃　〃　〃　〃　〃

VII. (3 × 3)

(ⅰ)	
(ⅱ)	
(ⅲ)	

授業・テストの感想