Java講座第3回

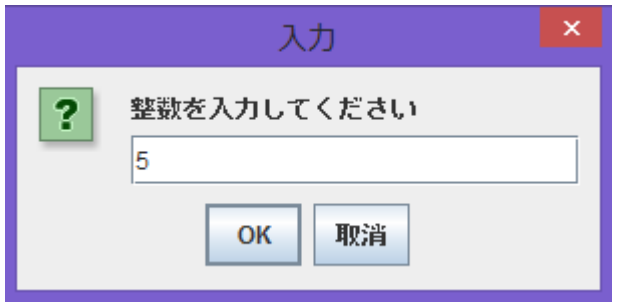
担当：大橋　山崎

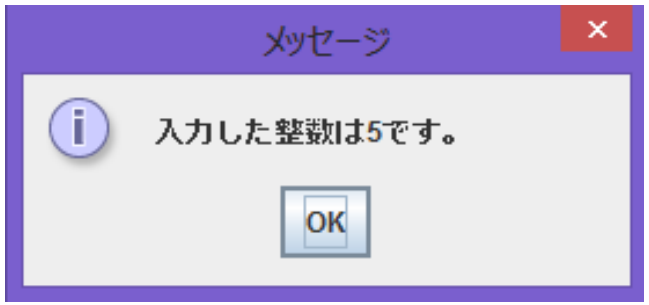
復習　ひな型

```
1  package k3;
2
3  public class Review {
4      public static void main(String[] args) {
5          new Review().start();
6      }
7
8      void start() {
9
10     }
11 }
```

復習　入出力

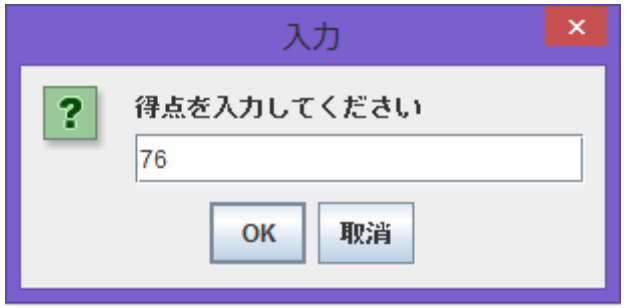
```
1  package k3;
2
3  import javax.swing.JOptionPane;
4
5  public class Review {
6      public static void main(String[] args) {
7          new Review().start();
8      }
9
10     void start() {
11         String input = JOptionPane.showInputDialog("整数を入力してください");
12         int num = Integer.parseInt(input);
13         JOptionPane.showMessageDialog(null, "入力した整数は" + num + "です。");
14     }
15 }
```

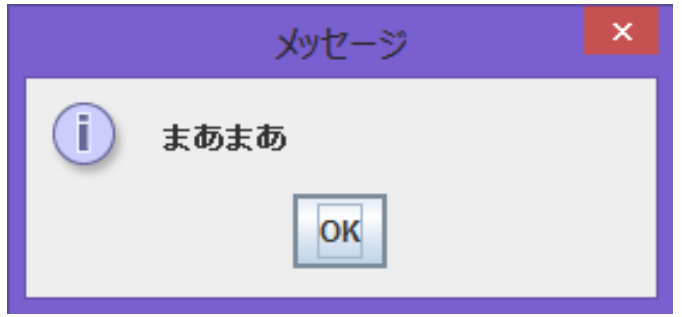




復習　if文

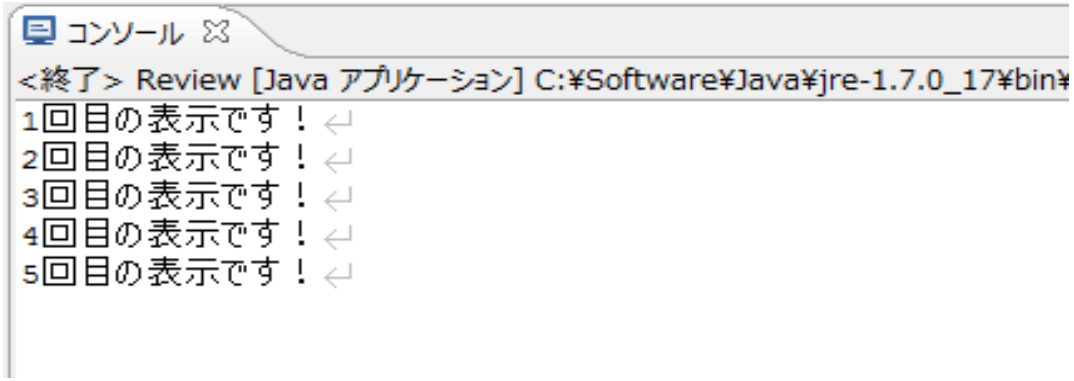
```
10  void start() {
11      String input = JOptionPane.showInputDialog("得点を入力してください");
12      int num = Integer.parseInt(input);
13      if (num >= 90) {
14          JOptionPane.showMessageDialog(null, "すばらしい");
15      } else if (num >= 80) {
16          JOptionPane.showMessageDialog(null, "すごい");
17      } else if (num >= 70) {
18          JOptionPane.showMessageDialog(null, "まあまあ");
19      } else if (num >= 60) {
20          JOptionPane.showMessageDialog(null, "もうちょっと頑張りましょう");
21      } else {
22          JOptionPane.showMessageDialog(null, "う～ん");
23      }
24  }
```





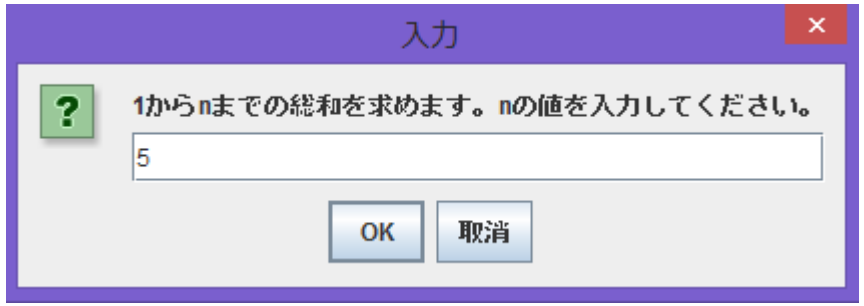
復習　for文

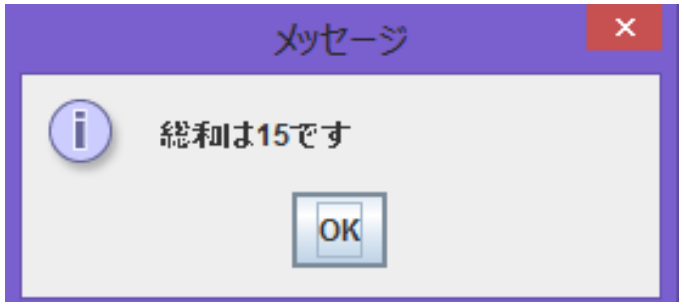
```
1  package k3;
2
3  public class Review {
4      public static void main(String[] args) {
5          new Review().start();
6      }
7
8      void start() {
9          for (int i = 0; i < 5; i++) {
10             System.out.println( i + 1 + "回目の表示です！");
11         }
12     }
13 }
```



復習　while文

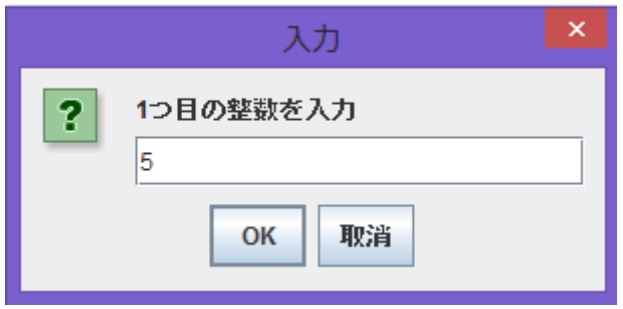
```
void start() {
    String input = JOptionPane.showInputDialog("1からnまでの総和を求めます。nの値を入力してください。");
    int num = Integer.parseInt(input);

    int sum = 0;

    while (num != 0) { //numが0以外の時ずっと続く
        sum += num;
        num--;
    }

    JOptionPane.showMessageDialog(null, "総和は" + sum + "です");
}
```

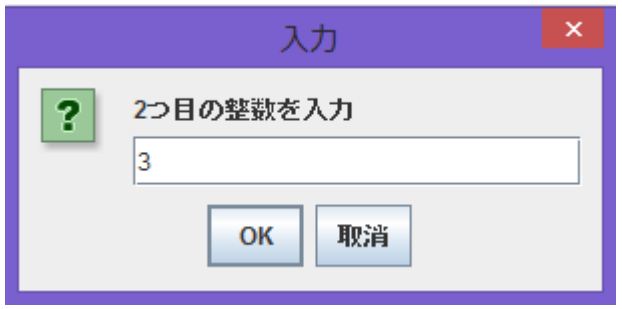


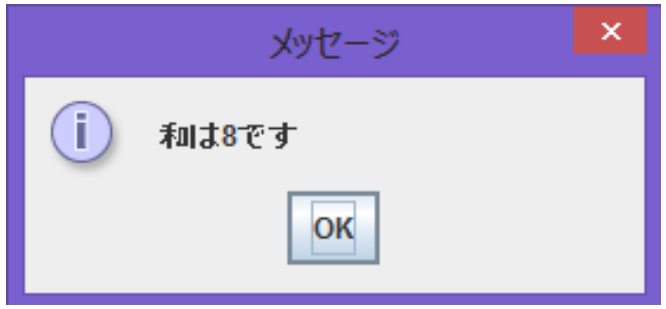


復習　メソッド

```
void start() {
    String input1 = JOptionPane.showInputDialog("1つ目の整数を入力");
    String input2 = JOptionPane.showInputDialog("2つ目の整数を入力");
    int num1 = Integer.parseInt(input1);
    int num2 = Integer.parseInt(input2);
    int result = Sum(num1, num2);
    JOptionPane.showMessageDialog(null, "和は" + result + "です");
}

int Sum(int n, int m) {
    int sum = n + m;
    return sum;
}
```

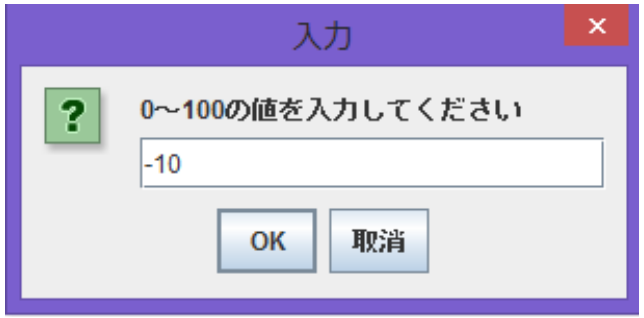


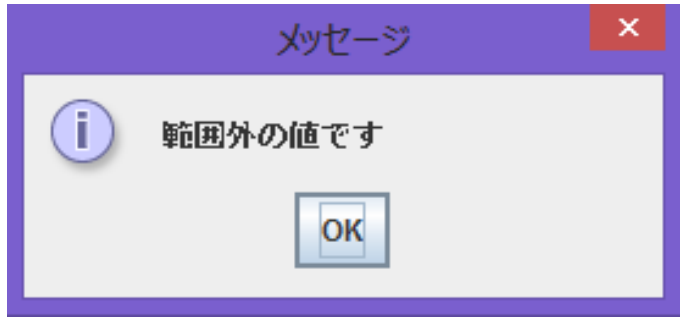




復習　boolean型のメソッド

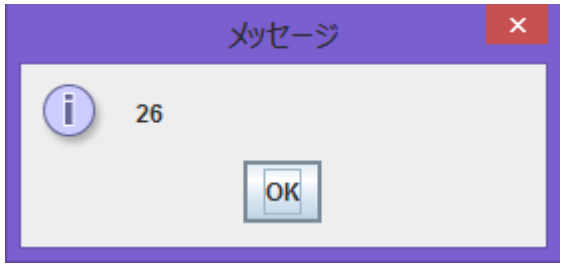
```
void start() {
    String input = JOptionPane.showInputDialog("0～100の値を入力してください");
    int num = Integer.parseInt(input);
    if (Cheak(num)) {
        JOptionPane.showMessageDialog(null, "正当な値です");
    } else {
        JOptionPane.showMessageDialog(null, "範囲外の値です");
    }
}

boolean Cheak(int num) {
    if (num >= 0 && num <= 100) {
        return true;
    } else {
        return false;
    }
}
```





復習　配列

```
void start() {
    int[] num = new int[5];
    num[0] = 5;
    num[1] = 2;
    num[2] = 8;
    num[3] = 1;
    num[4] = 10;
    int sum = 0;
    for (int i = 0; i < num.length; i++) { //num.lengthはnumの配列の長さを取る
        sum += num[i];
    }
    JOptionPane.showMessageDialog(null, sum);
}
```



```
void start() {
    String input = JOptionPane.showInputDialog("何個入力しますか？");
    int count = Integer.parseInt(input);

    // 入力した個数の配列を作成
    int[] num = new int[count];

    // 配列にデータを入れる
    for (int i = 0; i < num.length; i++) {
        String n = JOptionPane.showInputDialog(i + 1 + "番目の値を入力してください");
        num[i] = Integer.parseInt(n);
    }

    // 配列をメソッドSumに渡す      resultに返ってくる値が入る
    int result = Sum(num);

    JOptionPane.showMessageDialog(null, "合計は" + result + "です");
}

// 渡された配列の合計を計算し、結果を返す
int Sum(int n[]) {
    int sum = 0;
    for (int i = 0; i < n.length; i++) {
        sum += n[i];
    }
    return sum;
}
```

復習　対話型プログラム

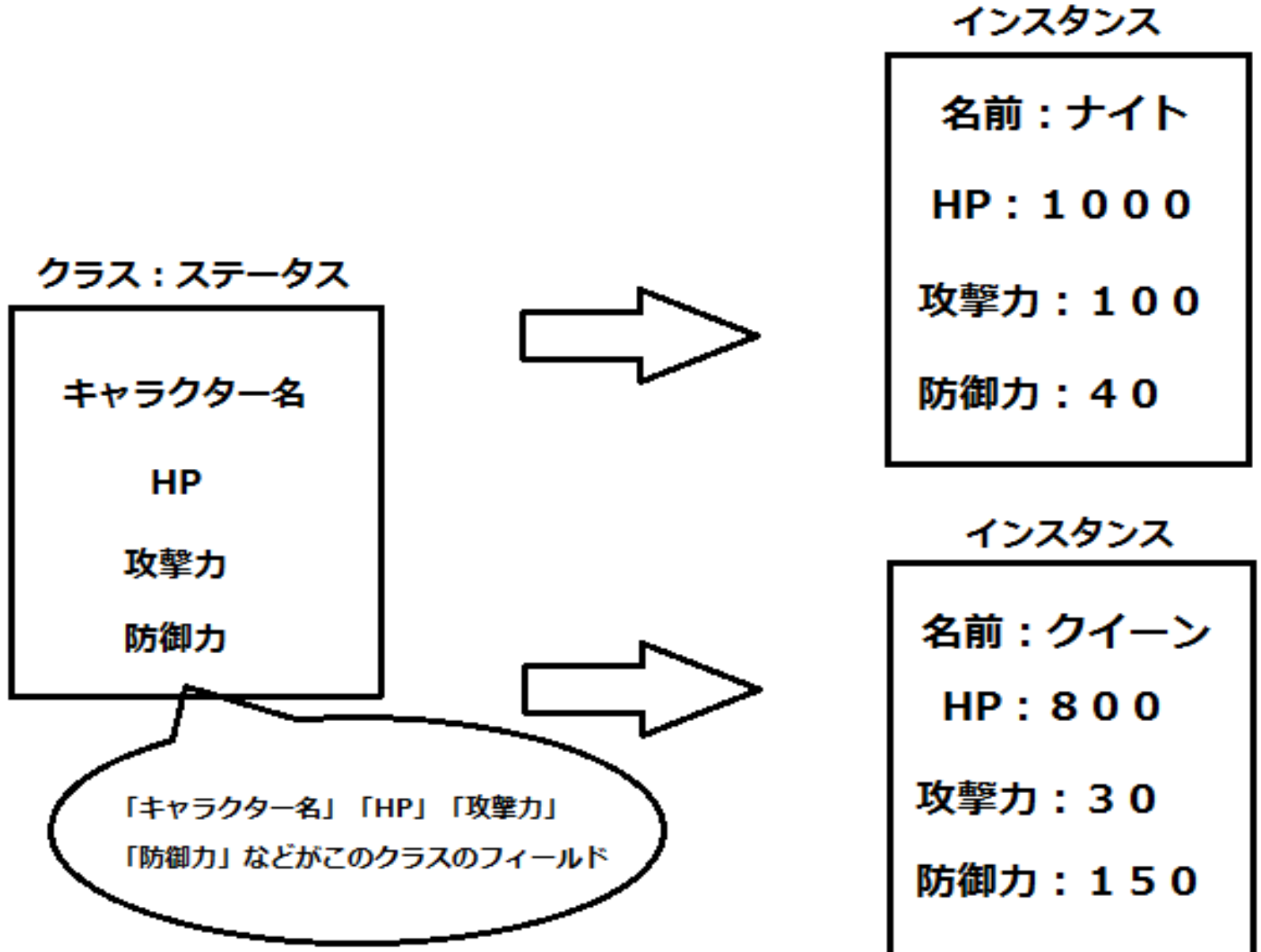
```
void start() {
    int sum = 0;

    while (true) { //breakしない限りずっと続く
        String input = JOptionPane.showInputDialog("整数を入力してください。0を入れると終了します。");
        int num = Integer.parseInt(input);

        if (num == 0) {
            JOptionPane.showMessageDialog(null, "終了します");
            break;
        }

        sum += num;
        JOptionPane.showMessageDialog(null, "今までの合計は" + sum + "です");
    }
}
```

クラスとインスタンス

クラス：複数のデータの形式をまとめたもの

フィールド：そのクラスの「性質」をあらわすもの
変数を使ってあらわす

インスタンス：クラスで定めた形式と同じデータを組み合わせた
データのこと


インスタンス
名前：ナイト
HP：１０００
攻撃力：１００
防御力：４０
クラス：ステータス
キャラクター名
HP
攻撃力
防御力
インスタンス
名前：クイーン
HP：８００
攻撃力：３０
防御力：１５０
「キャラクター名」「HP」「攻撃力」
「防御力」などがこのクラスのフィールド

クラスとフィールドの宣言

class <クラス名> {
　　<フィールドの型> <フィールドの名前>;
}

```
class status {              //クラスの宣言
    String name;            //フィールドの宣言
    int hp;                 //フィールドの宣言
    double attack;          //フィールドの宣言
    double defend;          //フィールドの宣言
}
```

クラスを書くときは、プログラムの最後に書いといてください。

```
package k3;

public class Review {
    public static void main(String[] args) {
        new Review().start();
    }

    void start() {

    }
}


class status {          //クラスの宣言
    String name;        //フィールドの宣言
    int hp;             //フィールドの宣言
    double attack;      //フィールドの宣言
    double defend;      //フィールドの宣言
}
```

インスタンスの生成

<クラスの名前> <変数の名前> = new <クラスの名前>( );

左辺でそのクラスの型の変数を生成し、右辺でそのクラスのインスタンスを生成してる。

インスタンスの生成だけなら右辺だけでOK！
ただインスタンスを生成して何もしないことはあまりないのでいきなり変数に代入させている。

まとめるとこんな感じ

```
package k3;

public class Review {
    public static void main(String[] args) {
        new Review().start();
    }

    void start() {
        //インスタンスを生成して、status型の変数に代入してる
        status player1 = new status();
    }
}


class status {          //クラスの宣言
    String name;        //フィールドの宣言
    int hp;             //フィールドの宣言
    double attack;      //フィールドの宣言
    double defend;      //フィールドの宣言
}
```

フィールドの利用

生成したインスタンスにはまだ具体的な値が設定されていないので設定する。

具体的には、

<インスタンスを指す変数名> . <フィールド名> = 具体的な値 ;

で設定できる。これを「フィールドに値を代入する」という。

こんな感じ

```
void start() {
    // インスタンスを生成して、status型の変数に代入してる
    status player1 = new status();

    player1.name = "ナイト";
    player1.hp = 1000;
    player1.attack = 100;
    player1.defend = 40;
}
```

2つ以上のインスタンスを生成するのも簡単

```
void start() {
    // インスタンスを生成して、status型の変数に代入してる
    status player1 = new status();
    status player2 = new status();

    player1.name = "ナイト";
    player1.hp = 1000;
    player1.attack = 100;
    player1.defend = 40;

    player2.name = "クイーン";
    player2.hp = 800;
    player2.attack = 30;
    player2.defend = 150;
}
```

クラスを用いたプログラムの例

```
void start() {
    status player1 = new status(); // インスタンスの生成
    status player2 = new status(); // インスタンスの生成

    player1.name = "ナイト";
    player1.hp = 1000;
    player1.attack = 100;
    player1.defend = 40;

    player2.name = "クイーン";
    player2.hp = 800;
    player2.attack = 30;
    player2.defend = 150;

    showStatus(player1); // インスタンスも引数に使えます
    showStatus(player2); // 同上
}

void showStatus(status player) {
    JOptionPane.showMessageDialog(null, "キャラ名 : " + player.name + "\nHP : "
            + player.hp + "\n攻撃力 : " + player.attack + "\n防御力 : "
            + player.defend);
}
```

```
class status {          // クラスの宣言
    String name;        // フィールドの宣言
    int hp;             // フィールドの宣言
    double attack;      // フィールドの宣言
    double defend;      // フィールドの宣言
}
```

演習１

Personクラスを作成して、2人のプロフィール(名前、年齢、身長、体重）を表示させなさい。

実行するクラス名「Q1」

・「name」、「age」、「height」、「weight」の４つのフィールドを作成
・インスタンスのそれぞれの値は自由
・showProfileメソッドを作成、使用すること

演習1　答え

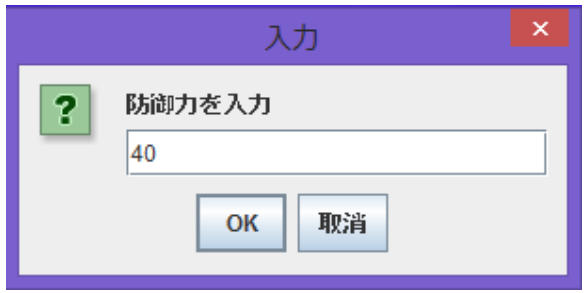
```
    void start() {
        Person person1 = new Person();
        Person person2 = new Person();

        person1.name = "Mike";
        person1.age = 25;
        person1.height = 175.5;
        person1.weight = 65.2;

        person2.name = "Alice";
        person2.age = 28;
        person2.height = 152.2;
        person2.weight = 45.1;

        showProfile(person1);
        showProfile(person2);
    }

    void showProfile(Person person) {
        JOptionPane.showMessageDialog(null, "名前： " + person.name + "\n年齢： "
                + person.age + "\n身長： " + person.height + "\n体重： "
                + person.weight);
    }
}

class Person {
    String name;
    int age;
    double height;
    double weight;
}
```

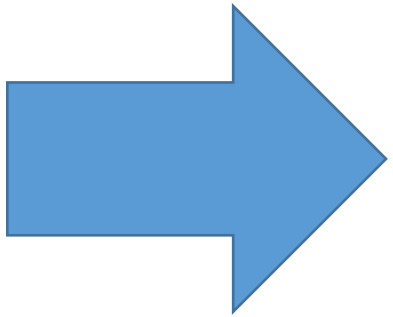
ちょっと難しいプログラム

```
void start() {
    status2[] characters = new status2[2];

    for (int i = 0; i < characters.length; i++) {
        characters[i] = inputStatus();
    }

    for (int i = 0; i < characters.length; i++) {
        showStatus(characters[i]);
    }
}

status2 inputStatus() {
    String input_name = JOptionPane.showInputDialog("名前を入力");
    String input_hp = JOptionPane.showInputDialog("HPを入力");
    String input_attack = JOptionPane.showInputDialog("攻撃力を入力");
    String input_defend = JOptionPane.showInputDialog("防御力を入力");
    status2 character = new status2(); // インスタンスを生成
    character.name = input_name;
    character.hp = Integer.parseInt(input_hp);
    character.attack = Double.parseDouble(input_attack);
    character.defend = Double.parseDouble(input_defend);
    return character;
}

void showStatus(status2 character) {
    JOptionPane.showMessageDialog(null, "名前 : " + character.name
            + "\nHP : " + character.hp + "\n攻撃力 : " + character.attack
            + "\n防御力 : " + character.defend);
}
```

```
class status2 {
    String name;
    int hp;
    double attack;
    double defend;
}
```

前のページのプログラムを実行すると・・・　(実際は2人表示する)





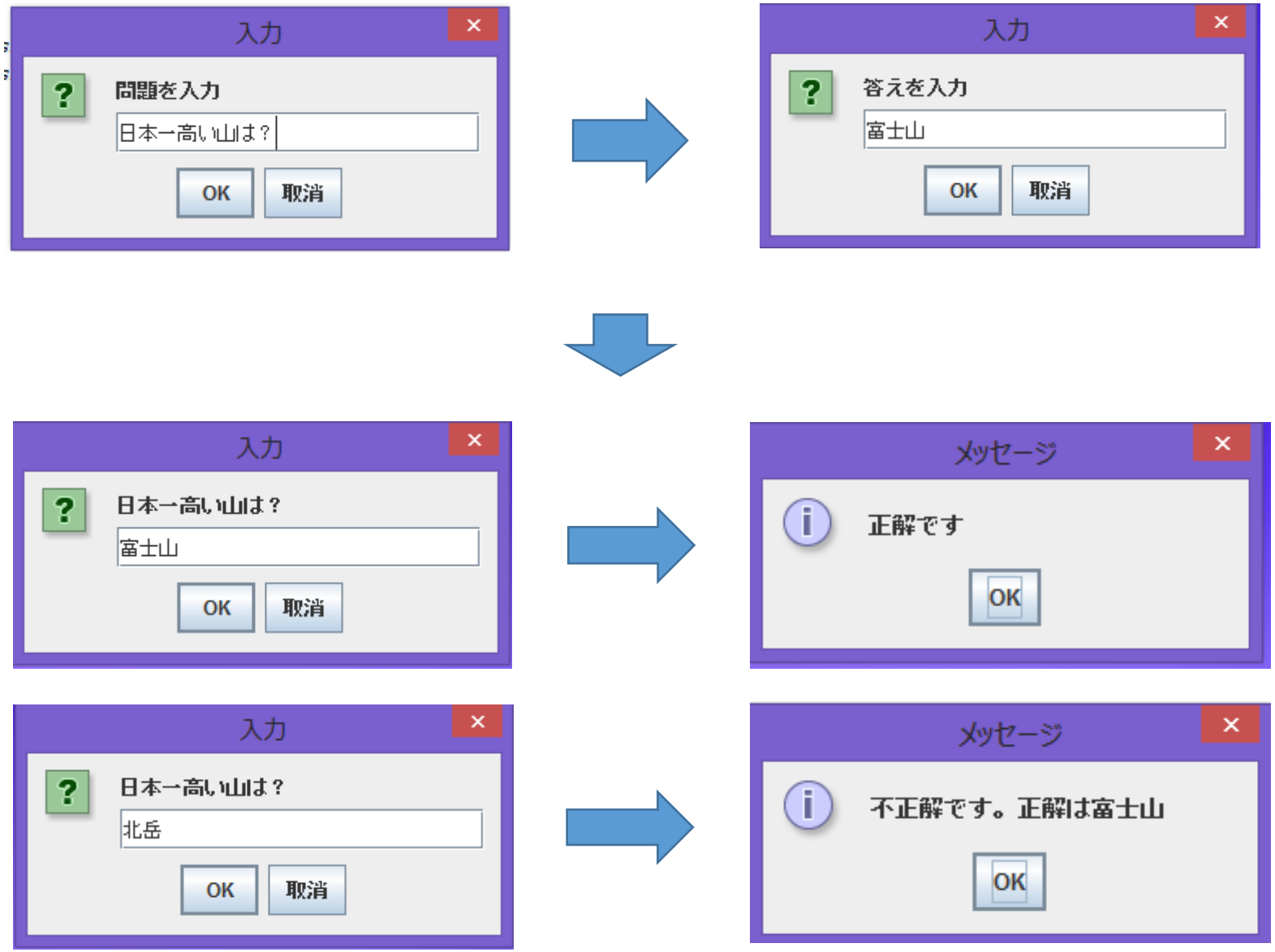
演習２

Questionクラスを作成し、２つの問題とその答えを自分で作成できるようにせよ。さらにその問に解答し、正誤を判断し結果（当たったかどうか）を出力せよ。

実行するクラス名 「Q2」

・「question」、「answer」の２つのフィールドを作成
・２つ前のスライドを少し変えればできる！

演習2の実行結果　　これが2回ずつ出ます



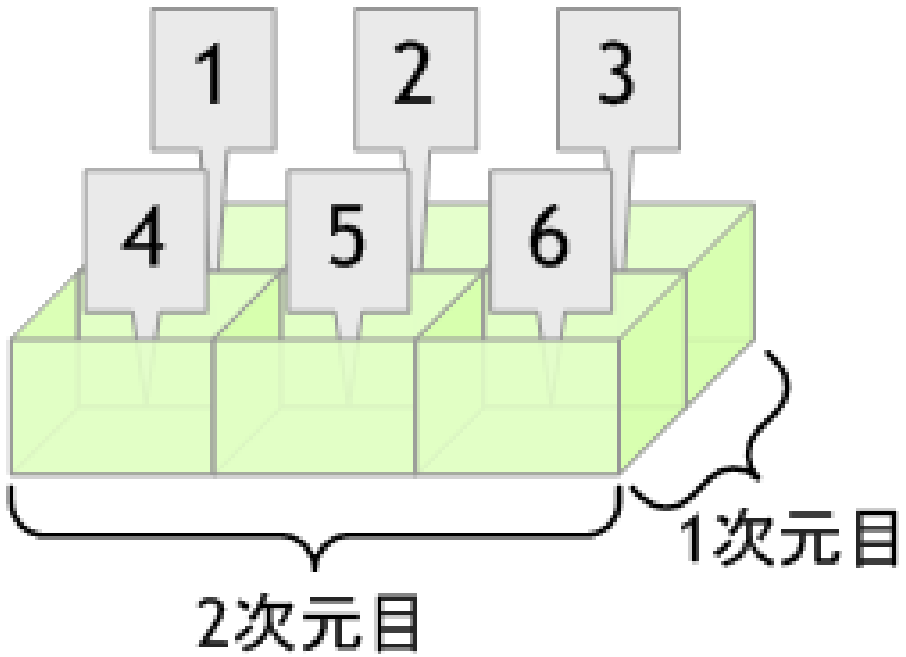
演習2の答え

```
void start() {
    Question[] q = new Question[2];

    for (int i = 0; i < q.length; i++) {
        q[i] = inputQuestion();
    }

    for (int i = 0; i < q.length; i++) {
        showQuestion(q[i]);
    }
}

Question inputQuestion() {
    String input_q = JOptionPane.showInputDialog("問題を入力");
    String input_a = JOptionPane.showInputDialog("答えを入力");
    Question q = new Question();
    q.quetion = input_q;
    q.answer = input_a;
    return q;
}

void showQuestion(Question q) {
    String input = JOptionPane.showInputDialog(q.quetion);
    if (input.equals(q.answer)) {
        JOptionPane.showMessageDialog(null, "正解です");
    } else {
        JOptionPane.showMessageDialog(null, "不正解です。正解は" + q.answer);
    }
}
```

```
class Question {
    String quetion;
    String answer;
}
```

２次元配列

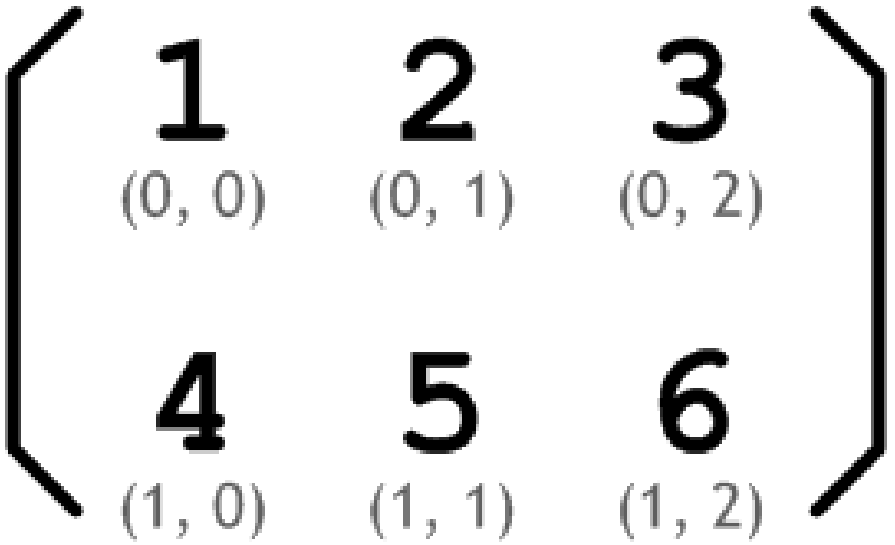
インデックス2つで、要素が1つ取り出せるような配列



使い方

```
int [ ] [ ] matrix  =  new  matrix[2] [3] ;
```

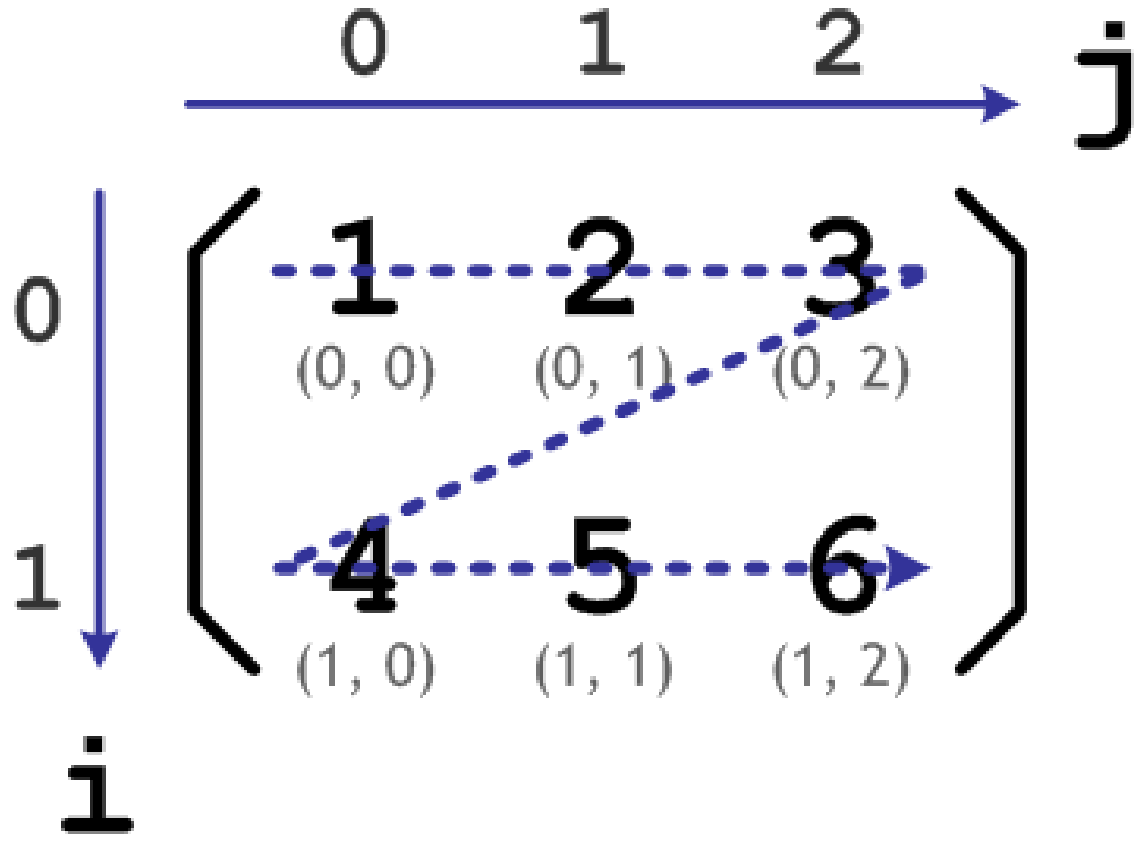
これで２×３の２次元配列が生成できる。（行列っぽい）



使用例

```
void start() {
    int[][] matrix = new int[2][3];
    matrix[0][0] = 1;
    matrix[0][1] = 2;
    matrix[0][2] = 3;
    matrix[1][0] = 4;
    matrix[1][1] = 5;
    matrix[1][2] = 6;
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 3; j++) {
            JOptionPane.showMessageDialog(null, "(" + i + "," + j + ") = "
                    + matrix[i][j]);
        }
    }
}
```

さっきのプログラムはこんな順番に表示してます
ｉを0で固定してjを0～2に動かし、次にｉを1で固定してjを0～2に動かす。

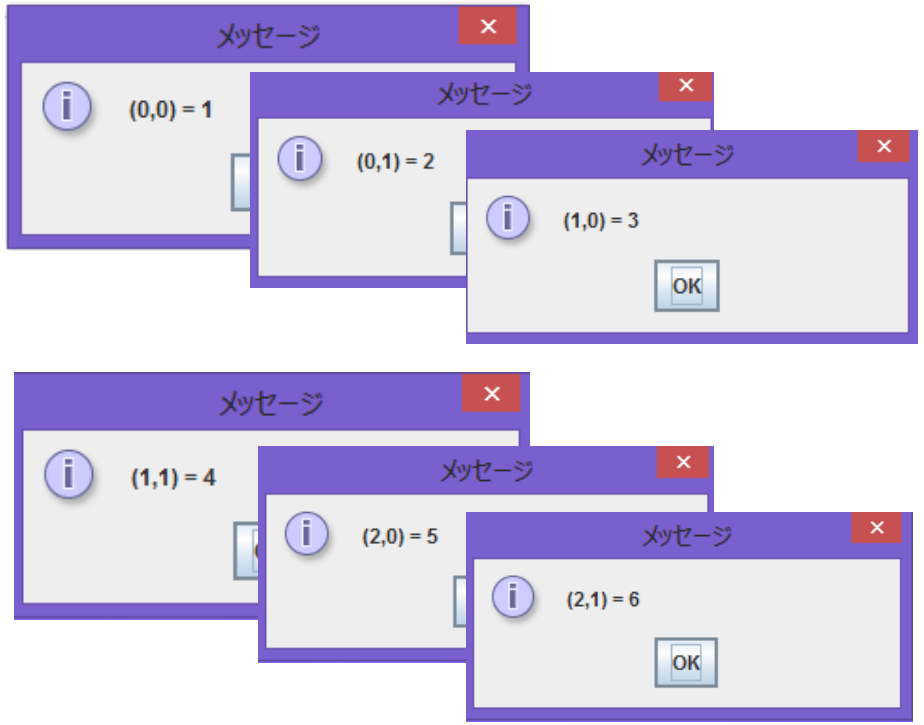


演習３

２次元配列を用いて、次の２次元配列のインデックスと要素を順番に表示せよ。

クラス名「Matrix」

1 (0,0)	2 (0,1)
3 (1,0)	4 (1,1)
5 (2,0)	6 (2,1)



演習3の答え

```
void start() {
    int[][] matrix = new int[3][2];
    matrix[0][0] = 1;
    matrix[0][1] = 2;
    matrix[1][0] = 3;
    matrix[1][1] = 4;
    matrix[2][0] = 5;
    matrix[2][1] = 6;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 2; j++) {
            JOptionPane.showMessageDialog(null, "(" + i + "," + j + ") = "
                    + matrix[i][j]);
        }
    }
}
```

お疲れ様でしたm(_ _)m