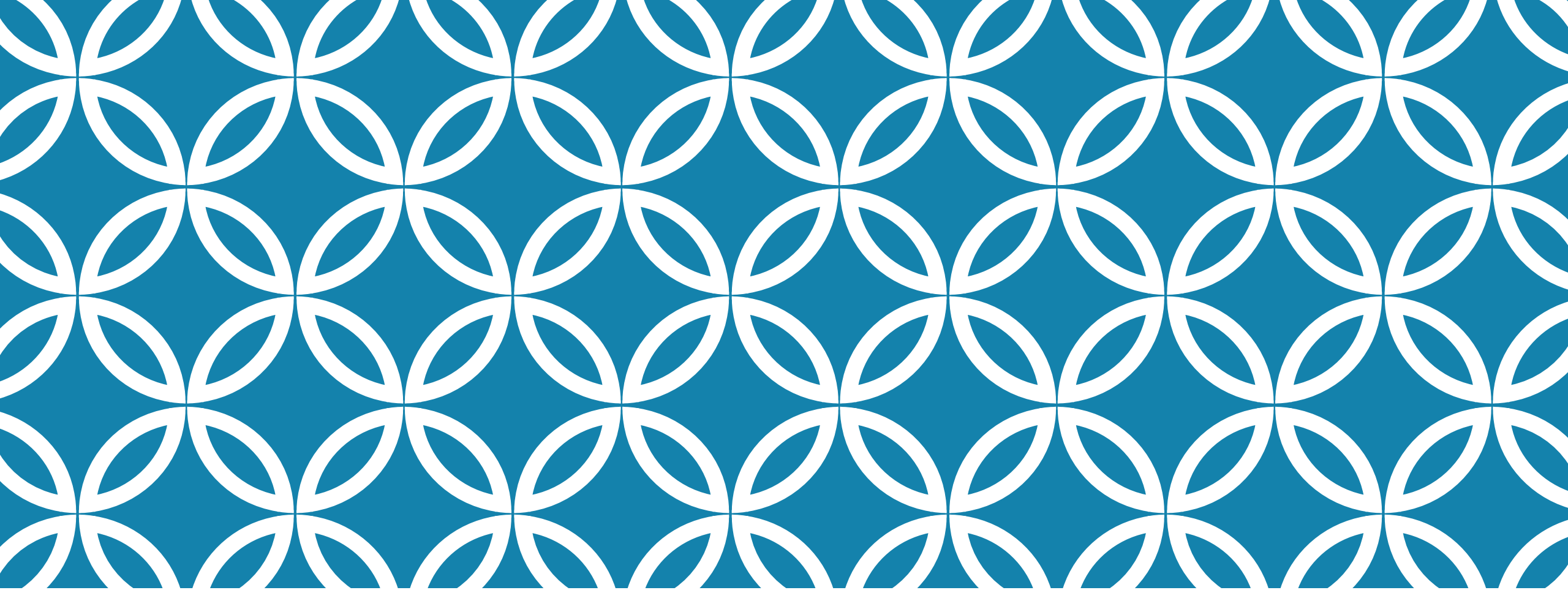


JAVA講座第2回

メソッド、配列、
対話型プログラミング
担当：海老原、小林

復習
任意の数値を5回入力させ、その総和を表示させよ

実行結果

解答

```
package java2014;

import javax.swing.JOptionPane;

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

    void start() {
        int result = 0;
        for (int i = 0; i < 5; i++) {
            int num = Integer.parseInt(JOptionPane.showInputDialog((i + 1)
                    + "番目の数値を入力してください"));
            result = result + num;
        }
        JOptionPane.showMessageDialog(null, "総和は" + result);
    }
}
```

これ以降ひな形を省略して表示させている部分があります

メソッド

指定した形式で値を返すもの
int型(整数),double型(実数),boolean型(論理値)などがあり、値を必ず返さなくてはならない

※論理値はtrue,falseという２つがあり、それぞれ条件式が満たされているかどうかをあらわす

voidは値を返さないメソッドにつける

例 1

```
package java2014;

import javax.swing.JOptionPane;

public class Ex1 {
    public static void main(String[] args) {
        new Ex1().start();
    }

    void start() {
        String message = MessageCreate();
        JOptionPane.showMessageDialog(null, message);
    }

    String MessageCreate() {
        String message = JOptionPane.showInputDialog("メッセージを入力してください");
        return message;
    }
}
```

String型メソッドの起動

return （変数）
で値を返す

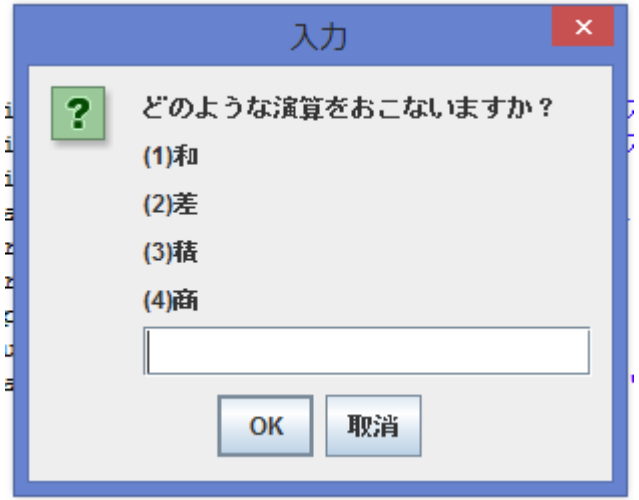
引数を利用したメソッド

```
void start() {
    String num = JOptionPane.showInputDialog("数値を入力");
    int a = Integer.parseInt(num);
    int b = num2(a);
    JOptionPane.showMessageDialog(null, b);
}

int num2(int n) {
    return 2 * n;
}
```

aを実引数としてnum2に渡している

演習１
２つの実数を入力させ、その四則演算を一つだけ指定し、計算結果を表示させよ
計算結果はメソッドを使用し返すこと

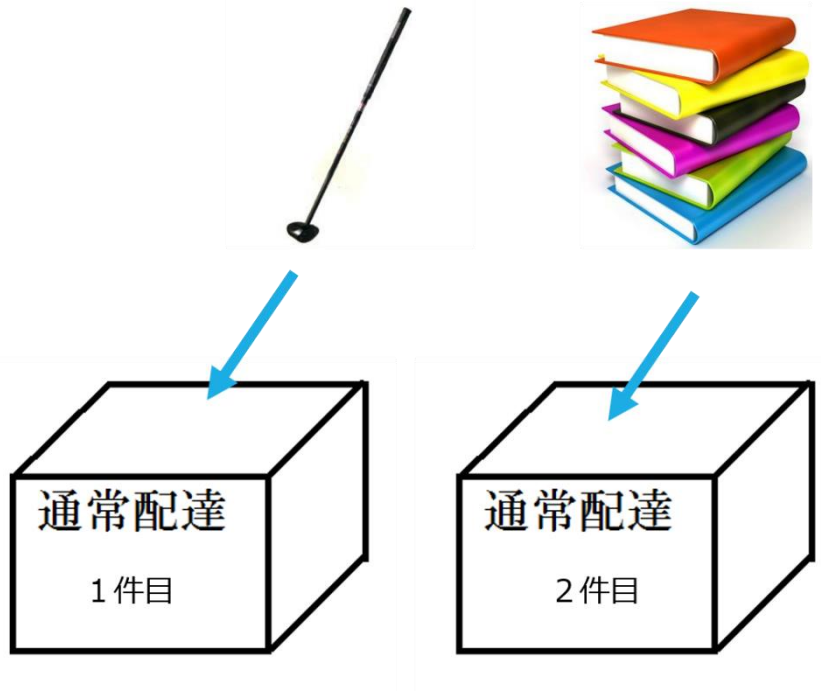


演算を指定するときには演算子に番号を振って、その値を入力することで選択させるとよい

配列

データをまとめる連なる枠組み

宅急便のように種類と順番が決められている



・配列は初めに型と大きさを宣言して作成する

・配列における番号は０から数えられる
たとえば大きさ５の配列の場合には０,１,２,３,４と番号が振られている

・配列の大きさ以上のデータを使用するとエラーになる

配列の作り方

型[]　名前＝　new　型[大きさ]；宣言

名前[番号]=データの中身；

配列を作成後、名前[番号]を今までの変数と同様に扱える

例２　String型の配列　Messagesを作製し、表示する

```
    void start() {
        String[] Messages = new String[4];
        Messages[0] = "おはよう";
        Messages[1] = "こんにちは";
        Messages[2] = "こんばんは";
        Messages[3] = "さようなら";
        JOptionPane.showMessageDialog(null, Messages[0]);
        JOptionPane.showMessageDialog(null, Messages[1]);
        JOptionPane.showMessageDialog(null, Messages[2]);
        JOptionPane.showMessageDialog(null, Messages[3]);
    }
}
```

演習２

任意の大きさの整数型配列を作製し、その後指定した番号に入っている数値未満の場所を数え上げ表示させよ

実行例　大きさ４　3,4,6,1　指定場所２番目

対話型プログラム

今までのプログラムは入力後に出力結果を表示して終了するものだった

対話型プログラムでは出力に対してユーザーがさらなる入力をするプログラムを作成する

While文を用いて作る

例３　指定した値が入力されるまで入力を求める

```
void start() {
    boolean check1 = true;
    int x = 10;
    while (check1) {
        String input = JOptionPane.showInputDialog("整数を入力してください");
        int num = Integer.parseInt(input);
        if (num == x) {
            JOptionPane.showMessageDialog(null, "指定の値が入力されました");
        } else {
            JOptionPane.showMessageDialog(null, "残念ながらはずれです。指定の値との差は
                    + Gap(num, x) + "です");
        }
        check1 = check(num, x);
    }
}

boolean check(int num, int x) {
    if (num == x) {
        return false;
    } else {
        return true;
    }
}

int Gap(int num, int x) {
    if (x > num) {
        return x - num;
    } else {
        return num - x;
    }
}
```

while文の条件式（初期化する）

While文の繰り返しを行うか判定する

指定した値と入力した値の差を返す

演習３

ユーザーとCPUが互いに３以下の数値を足していき、１０以上となった方が負けとなるゲームを作製せよ。
勝敗が決まるまで入力を求め、範囲外の数値では処理を行わないように注意せよ。またCPUは合計が１０未満となる入力が可能な場合にはその数値で入力を行い、ユーザーの敗北を狙える場合には（合計が９となる場合）そのような入力をさせよ