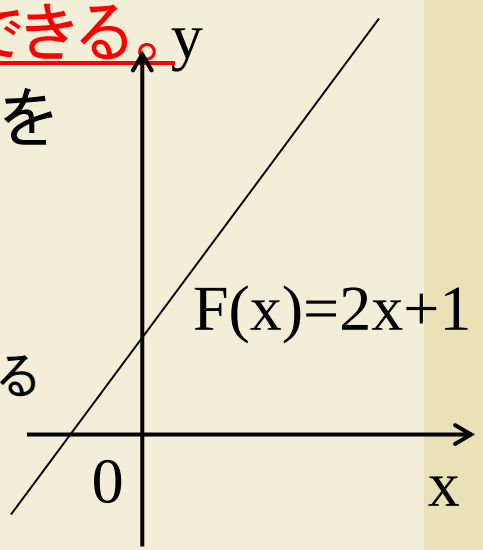
Method(C言語では関数と呼ぶ)

- メソッドを使うと、処理を纏めて管理することができる
  → 処理(メソッド)の再実行（再利用）が簡単にできる。
- 元々はC言語の関数であり、入力値に対する値を定義するもの
  - 数学では、F(x) = 2x + 1　など

    ↑入力値(引数) x が決まれば、F(x)が決まる



- これを応用して、複雑な処理も、外面的にはひと固まりの処理として扱う
  - 入力値に対する処理をして、その結果値を返す
- 関数の書式(戻り値が無い場合は　void型として指定する)

```
戻り値のデータ型　関数名（引数のデータ宣言の並び）
{
        …
        return(戻り値);
}
```

例

```
int  func1(int x){
        x = 2*x + 1;
        return x ;
     }
```

演習：メソッドの利用例

```
import java.math.*;
class OmikujiFunc {
    public String omikuji(String args[]) {
        double  randomNumber = Math.random();
        String result = "";
        if      (randomNumber < 0.2) { result = "凶です";  }
        else if(randomNumber < 0.4) { result = "小吉です";}
        else                                { result = "大吉です";}
        return result;
    }
    public static void main(String args[]) {
        String myResult= omikuji( );
        System.out.println(myResult );
    }
}
```

結果をメイン文に帰す(戻す)

Omikujiメソッドでおみくじの結果が得られる

おみくじの結果を表示する

```
c:¥Users¥user> cd c:¥src
c:¥src>notepad OmikujiFunc.java
c:¥src>javac  OmikujiFunc.java
c:¥src>java OmikujiFunc
```

課題３－１：メソッドの利用例

```
import java.math.*;
class OmikujiFunc {
    public String omikuji(String args[]) {
        double  randomNumber = Math.random();
        String result = "";
        if      (randomNumber < 0.2) { result = "凶です";  }
        else if(randomNumber < 0.4) { result = "小吉です";}
        ・・・
        else                                { result = "大吉です";}
        return result;
    }
    public static void main(String args[]) {
        String myResult= omikuji( );
        System.out.println(myResult );
    }
}
```

これを基にして、5種類のおみくじを左下のように自分の名前と、今日の・・という表示にさせなさい。

```
c:¥src>notepad OmikujiFunc.java
c:¥src>javac OmikujiFunc.java
c:¥src>java OmikujiFunc
こんにちは田中二郎さん。
今日の運勢は中吉です。
```

演習：メソッドの利用例

緑色の部分は決まっているもので、各自で黒い部分のみを考えて下さい

```
import java.math.*;
import java.util.Scanner;
class NumberGame{
    public static void myGame(){/* ゲームを実行する関数 */
        int result = (int) (Math.random()*10.0); /* random()は 0.0～1.0 の乱数を生成、*/
        Scanner data; int guess = 0;  /* 人間が推測した値を格納するための変数 */
        System.out.println("数当てゲーム  0～2のどれかを推測して下さい");

        while( guess != result ){ /* 推測された値が一致するまで実施する */
            data = new Scanner(System.in); guess = data.nextInt();/* キーボード入力 */
            if(guess==result)      { System.out.println("当たり"); }
            else if(guess>result) { System.out.println("もっと大きいか小さい数です"); }
            else                          {       }
        }
    }
    public static void main(String args[])  {
        myGame();
    }
}
```

どんな複雑な処理も、メソッドで定義しておけば、それを呼び出すだけで良い

```
c:¥src>notepad NumberGame.java
c:¥src>javac  NumberGame.java
c:¥src>java NumberGame
```

課題3－2：メソッドの利用例

数当てを0から30までとして、当たったら、当たり として、当たったら、当たり はずれだったら、もっと大きな数かもっと小さな数を入力 すれば良いかのヒントを表示させて下さい。

```
import java.math.*;
import java.util.Scanner;
class NumberGame{
    public static void myGame(){/* ゲームを実行する関数 */
        int result = (int) (Math.random()*？); /* random()は 0.0～1.0 の乱数を生成、resultに1から10の値が代入される */
        Scanner data; int guess = 0;  /* 人間が推測した値を格納するための変数 */
        System.out.println("数当てゲーム  0～2のどれかを推測して下さい");

        while( guess != result ){ /* 推測された値が一致するまで実施する */
            data = new Scanner(System.in); guess
            if(guess==result)      { System.out.pri
            else if(guess>result) { System.out.prin
            else                        { System.out.pri
        }
    }
    public static void main(String args[])  {
        myGame();
    }
}
```

```
c:¥src>notepad NumberGame.java
c:¥src>javac  NumberGame.java
c:¥src>java NumberGame
15
もっと大きい数です
18
もっと小さい数です
16
おめでとうございます、当たりです
```

演習：メソッドの利用例

```
import java.math.*;
import java.io.*;

class SlotFunc {
    public int slot1( void ) {
        int number = (int) (Math.random()*3.0) ;
    }
    public int slot2( void ) {
        int number = (int) (Math.random()*3.0) ;
    }
    public static void main(String args[]) {
       System.out.print(" ["+ slot1( )+" ]");
       System.out.print(" ["+ slot1( )+" ]");
    }
}
```

結果をメイン文に帰す(戻す)

結果をメイン文に帰す(戻す)

Omikujiメソッドでおみくじの結果が得られる

おみくじの結果を表示する

```
c:¥Users¥user> cd c:¥src
c:¥src>notepad SlotFunc.java
c:¥src>javac  SlotFunc.java
c:¥src>java SlotFunc
```

演習：タイマーの利用例

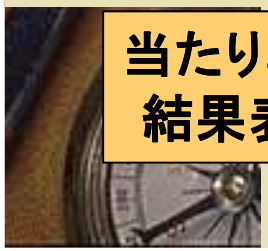
```
import java.math.*;
import java.io.*;
class MySlot {
    public static void main(String args[]) {
        System.out.println( "スロットマシーンスタート" );
     int slot[]= new int[5];     //スロットの各列の数値の保存用配列
        for(int i=0; i<2; i++ ){     //3つのスロットを順次止める
            slot[i] = (int) (Math.random()*3.0) ;  //ここでスロットの値が決まる
            try{Thread.sleep(1000);}catch(Exception e){ }
            System.out.print("「" + slot[i] + "」" );
        }
        System.out.println();
        if( (slot[0]==slot[1])&&(slot[1]==slot[0]) )System.out.println("当たり");
        else System.out.println("");
    }
}
```

```
c:¥Users¥user> cd c:¥src
c:¥src>notepad  MySlot.java
c:¥src>javac  MySlot.java
c:¥src>java MySlot
```

課題３－３：タイマーの利用例

これを基にスロットが3列で、各列の数が1から5の数のゲームに変えなさい。提出の際にはメール本文にプログラムの各行がどんな処理をしているのか、全て説明を書くこと

```
import java.math.*;
import java.io.*;
class MySlot {
    public static void main(String args[]) {
        System.out.println( "スロットマシーンスタート" );
        int slot[]= new int[5];     //スロットの各列の数値の保存用配列
        for(int i=0; i<2; i++ ){     //3つのスロットを順次止める
            slot[i] = (int) (Math.random()*3.0) ;  //ここでスロットの値が決まる
            try{Thread.sleep(1000);}catch(Exception e){ }
            System.out.print("「" + slot[i] + "」" );
        }
        System.out.println();
        if( (slot[0]==slot[1])&&(slot[1]==slot[0]) )System.out.println("当たり");
        else System.out.println("");
    }
}
```



当たりと、はずれを正しく表示させ、結果表示をこのように【と】でくくる

```
c:¥src>notepad  MySlot.java
c:¥src>javac  MySlot.java
c:¥src>java MySlot
スロットマシーンスタート
【4】【3】【1】
```