# 第2講　オブジェクトの基礎(1)

- 手続き型プログラミングとオブジェクト指向プログラミング
- 両者の相違点
- オブジェクトとは何か？
- クラスとは何か？
- オブジェクト指向とは？

## 手続き型プログラミングとオブジェクト指向プログラミング

- 手続き型プログラミング
  - コード
    - 関数や手続きの中に置かれる
    - 関数や手続きは入力と出力をもつブラックボックス
  - データ
    - データは別の構造に置かれ、関数や手続きにより操作される
  - 特徴
    - データが手続きから分離される
    - データはグローバルになるため、範囲外のデータをどこからでも容易に変更可能

      ⇒

      つまり、データにアクセスするコードを制限できないために、テストやデバックが困難

# 手続き型プログラム（C言語）

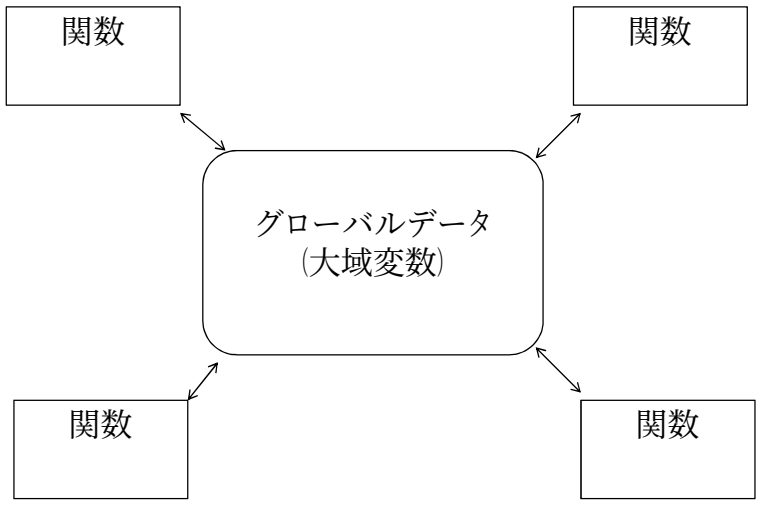
```
#include <stdio.h>
/*
 * メッセージの定義
 */
char message[] = "Hello World!";
/*
 * 関数の定義
 */
void printArg(int no, char *arg) {
    printf("argument %d is: %s¥n", no, arg);
}
/*
 * mainの定義
 */
void main(int argc, char *argv[]) {
    int i;
    for (i = 0; i < argc; i ++) {
        printArg(i,argv[i]);
    }
    printf("%s¥n", message);
}
```

・ヘッダファイルの宣言
　プログラム中で使うライブラリの宣言

・大域変数の定義
　プログラムのすべての関数から利用可能な変数を定義

・手続き（関数）の定義
　プログラムで利用する関数を定義

# 手続き型プログラミング



## 手続き型プログラミングとオブジェクト指向プログラミング

- オブジェクト指向プログラミング
  - 特徴
    - データと動作を結合して1つの完全なパッケージにすることで、手続き型プログラミングの問題を解決
  - オブジェクト
    - 整数や文字列などを保持するだけでなく、それらのデータ型を操作するメソッド（関数）も保持
    - オブジェクトのメンバへのアクセスを制限

      ⇒

      データ型やメソッドのメンバを他のオブジェクトから隠蔽

## オブジェクト指向プログラム（Java言語）

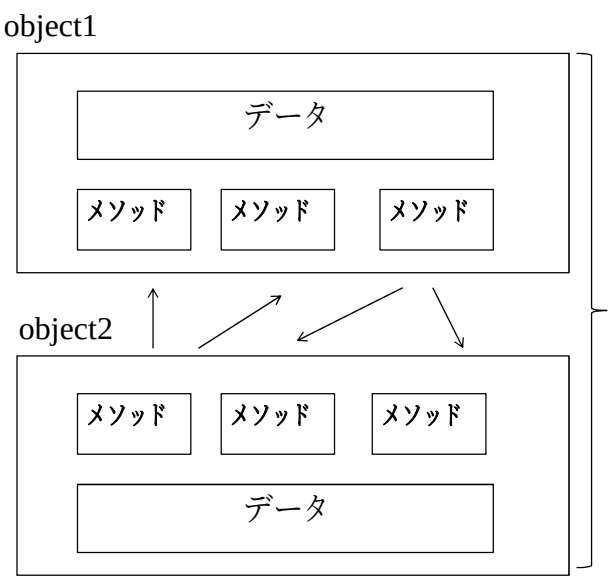
```
import java.lang.*;
import java.io.*;
import java.util.*;

class HelloWorld {
  public static void main(String[] argv) {
    System.out.println(”Hello World!”);
  }
}
```

・プログラムを構成するクラスの定義

・mainメソッドの定義

public static void main(String[] argv)（mainメソッド）を定義したクラスは1つ必要

# オブジェクト指向プログラミング




・グローバルデータ（大域変数）に相当するものはない
・関数定義は単体で存在しない

メッセージを介したオブジェクト間通信

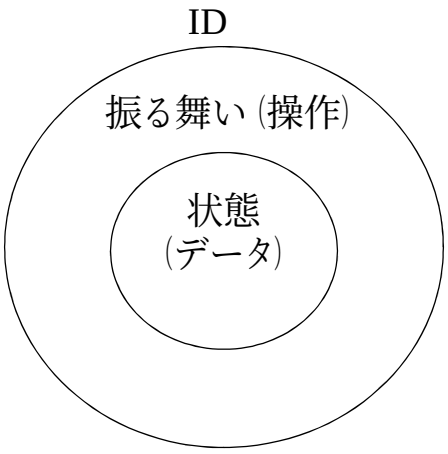
# オブジェクト指向とは

- 「オブジェクト」＝「モノ、物体」
- 「指向」＝「～を目指して向かうこと」
- 「オブジェクト指向」
  - モノ（物体）を目指して把握する考え方
  - モノ
    - コーヒー、スポーツカー、会社、大学、時間、恋愛
  - 目に見える物体、事象や概念を含めてすべてをオブジェクト（モノ）と捉える考え方

# オブジェクトとは何か？

- オブジェクト指向プログラムの構成単位
- プログラムはオブジェクトの集まり
- オブジェクト=データ（属性）＋振る舞い（操作）
- オブジェクトのデータ
  - オブジェクトに格納されたデータ → オブジェクトの状態を表現
  - 上記のデータを属性（プロパティー）と呼ぶ
- オブジェクトは、一意に識別できるアイデンティティー（識別子）を持つ。

# オブジェクト

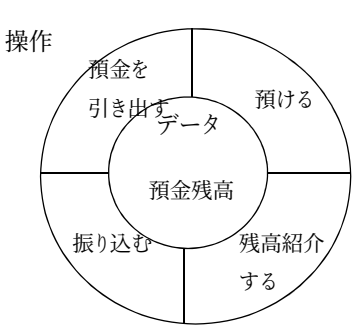


# オブジェクトとは何か？

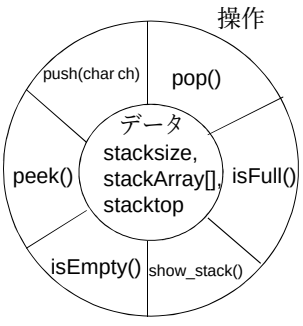
- オブジェクトの振る舞い
  - オブジェクトが何を行なえるかを示す
  - 手続き型言語
    - 手続き、関数、サブルーチン
  - オブジェクト指向プログラミング
    - メソッドにより表現
    - メソッドの呼び出しには、オブジェクトにメッセージを送信する
    - メソッドの利用には、次の情報が必要
      - メソッドの名前
      - メソッドに渡すパラメータ
      - メソッドの戻り値の型
- オブジェクトのインスタンス化
  - オブジェクトを生成すること

# オブジェクトの例

口座オブジェクト




CharStack オブジェクト



## クラスとは何か？

- オブジェクトの設計図
  - オブジェクトをインスタンス化（作成）するときに利用
- オブジェクト指向ソフトウエア
  - オブジェクトとクラスでは、クラスが最初に存在する
- クラスの定義
  - オブジェクトを抽象化した枠組み
  - データ
    - クラスのデータは属性により表現
  - メソッド（振る舞い）
    - クラスの振る舞いを実装
  - メッセージ
    - オブジェクト間の通信メカニズム

## オブジェクト指向とは

- プロパティ
  - オブジェクト指向の世界では，データはどこかのオブジェクトの性質を表わすものであり，オブジェクトの内部にカプセル化されている
  - データのかわりに，プロパティと呼ぶ
- メソッド
  - オブジェクト指向の世界では，プログラムもどこかのオブジェクトを操作する方法を表わすものであり，オブジェクトの内部にカプセル化されている
  - プログラムといわずに，メソッドと呼ぶ

# オブジェクト指向とは

- メッセージ
  - プログラムの起動は, オブジェクトに対してメソッドを動かすように依頼することにより実行
  - オブジェクトに対して, メッセージを動かすように依頼するものをメッセージを呼ぶ
- メッセージパッシング（オブジェクト間のやりとり）
  - オブジェクトにメッセージを送ること

# メッセージパッシングの例