PFCore(RTミドルウェア)トレーニング 中級編

10:00-11:00	第１部：RTコンポーネントプログラミングの概要
	担当：安藤慶昭（産業技術総合研究所）
	概要:RTコンポーネントの作成方法，設計時の注意点などの概要について解説します。
11:00-12:00	第２部：RTミドルウェア(PFcore)開発支援ツールとRTコンポーネントの作成方法
	担当：坂本 武志（株式会社 グローバルアシスト）
	概要:RTコンポーネントを開発するために必要なツールのインストール方法，標準ツールRTCBulderを使用して、RTコンポーネントを開発する方法の概略を説明します．
12:00-13:00	休憩
13:00-17:00	第３部：RTコンポーネント開発実習
	担当：安藤慶昭（産業技術総合研究所）
	概要:OpenRTM-aistでのコンポーネントの作成方法を実際に体験して頂きます．画像処理システムを対象にRTCBuilderを使用したRTコンポーネントの設計，実装を行います．

~~第１部　RTコンポーネントプログラミングの概要~~

第１部　プログラミング入門

（独）産業技術総合研究所
知能システム研究部門
安藤慶昭

概要

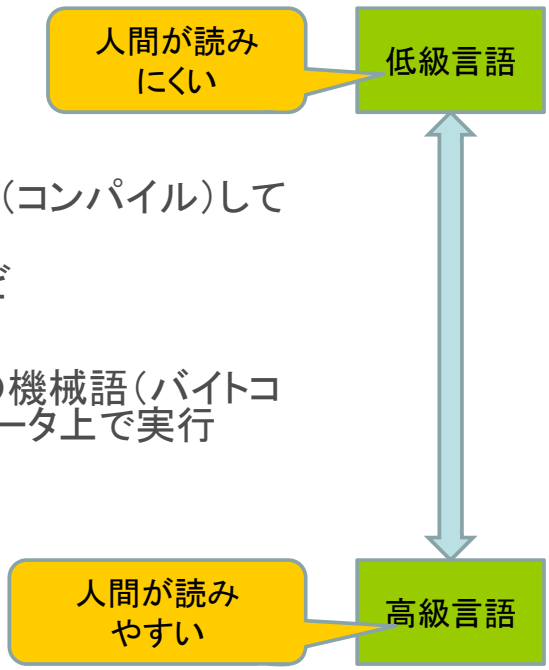
- プログラムとプログラミング言語
- プログラミング言語を学習するということ
- 実習（1）
- オブジェクト指向プログラミング
- 実習（2）

プログラム

- プログラムとは？
  - コンピュータにやらせたいことを書いた手順書
    - アルゴリズム
  - コンピュータは機械語（マシン語）しか理解できない
- プログラミング言語
  - プログラムはプログラミング言語で書く
  - プログラミング言語にはいろいろな種類がある
  - それぞれ長所・短所がある（適材適所）

プログラミング言語
（処理系による分類）

- 機械語
- アセンブリ言語
- コンパイル型言語
  - プログラムをすべて機械語に翻訳（コンパイル）してから実行
  - C、C++、FORTRAN、Pascal など
- 中間言語型
  - プログラムを仮想コンピュータ用の機械語（バイトコード）に翻訳してから仮想コンピュータ上で実行
  - Java、.NET（C#など）
- インタプリタ型
  - プログラムを一行ずつ実行
  - Ruby、Python、Perl、BASIC



プログラミングパラダイム

- オブジェクト指向プログラミング
- 関数型プログラミング
- ジェネリックプログラミング
- データ指向プログラミング
- 構造化プログラミング - 非構造化プログラミング
- 命令型プログラミング - 宣言型プログラミング
- メッセージ送信プログラミング - 命令型プログラミング
- 手続き型プログラミング - 関数型プログラミング
- 値レベルプログラミング - 関数レベルプログラミング
- 逐次実行型プログラミング - イベント駆動型プログラミング
- スカラプログラミング - ベクトルプログラミング
- クラスベースプログラミング - プロトタイプベースプログラミング - Mixin
- 制約プログラミング - 論理型プログラミング
- コンポーネント指向プログラミング
- アスペクト指向プログラミング
- パイプラインプログラミング
- 課題指向プログラミング
- リフレクティブプログラミング
- データフロープログラミング (スプレッドシート)
- ポリシーベースプログラミング
- ツリープログラミング
- 註釈プログラミング
- 属性指向プログラミング
- コンセプト指向プログラミング

プログラミング言語の特徴

	実行速度	リアルタイム実行	コンパイル	オブジェクト指向	手軽さ
C/C++	◎	◎	必要	△／○	△
Java	○	△	必要	◎	○
Python	×	×	不要	○	◎

	移植性	動的処理	GUIの作りやすさ	学習のしやすさ	大規模開発
C/C++	△	×	×	△／×	○／◎
Java	○	△	△	○	◎
Python	○	○	○	◎	△

用途・目的に応じて適切な
プログラミング言語を選択することが重要

プログラミング言語を学習するということ

- コンピュータに使われず、コンピュータを使う
- ツールやGUIベースのアプリケーションは定型処理しかできないが、プログラミング言語を駆使すれば意のままにできる
- さまざまな処理を自動化できる
- 何度でも同じ処理を行い、同じ結果を得ることができる
- 作業・処理の知見を残す・伝えることができる

プログラミング言語を学ぶ

- 文法を覚える
- コンパイル、実行の方法を覚える
- 関数やライブラリの使い方を覚える
  - いっぺんに覚える必要はない、少しずつ
- デバッグの方法を覚える

いろいろな言語のHello World

C++

```
#include <cstdlib>
#include <iostream>
int main ()
{
  std::cout << "Hello, world!" << std::endl;
  return EXIT_SUCCESS;
}
```

いろいろな言語のHello World

Python

```
print "Hello, world!“
```

（Version 3以降は）

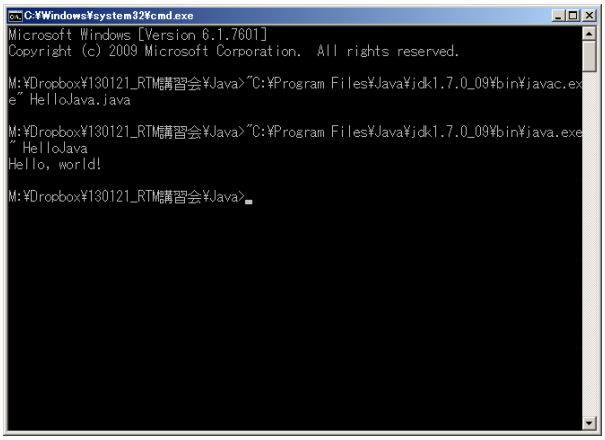
```
print(“Hello, world!”)
```


いろいろな言語のHello World

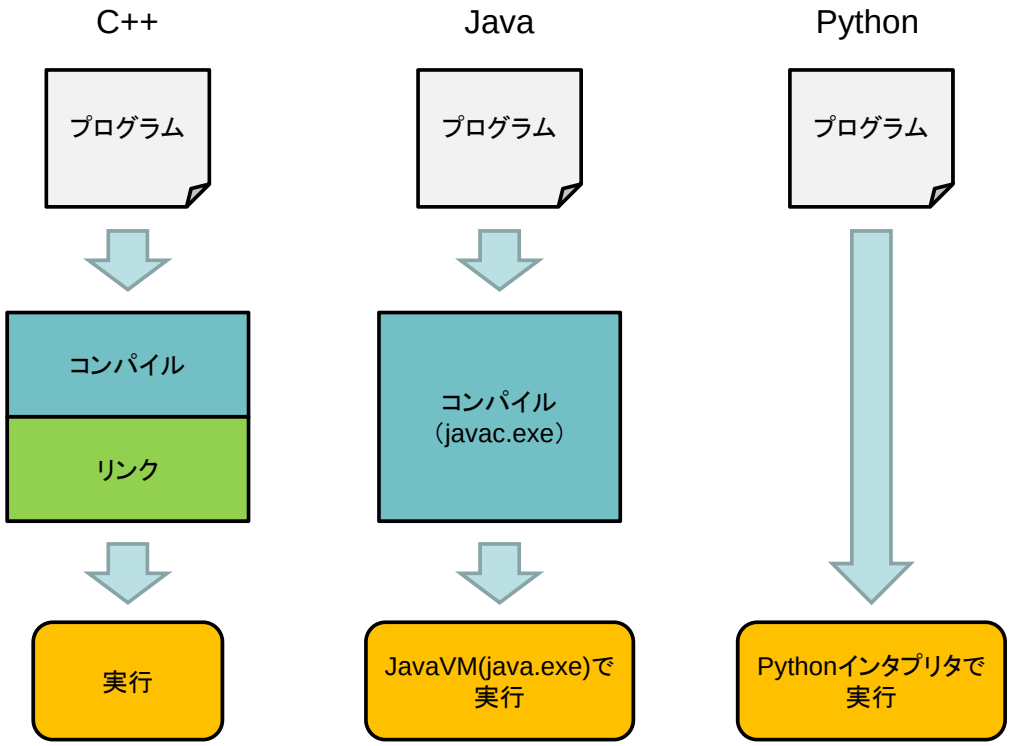
Java

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

public class HelloFrame extends Frame {
 HelloFrame(String title) {
  super(title);
 }
 public void paint(Graphics g) {
  super.paint(g);
  Insets ins = this.getInsets();
  g.drawString("Hello, World!", ins.left + 25, ins.top + 25);
 }
 public static void main(String[] args) {
  HelloFrame fr = new HelloFrame("Hello");

  fr.addWindowListener(
   new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
     System.exit(0);
    }
   }
  );
  fr.setResizable(true);
  fr.setSize(500, 100);
  fr.setVisible(true);
 }
}
```

```
class HelloJava {
  public static void main(String[] args) {
  System.out.println(“Hello, world!");
  }
}
```



プログラミングの流れ



実習（1）

- Hello Worldプログラムを作ってみる

http://d.hatena.ne.jp/arakik10/20100817/p2

オブジェクト指向プログラミング

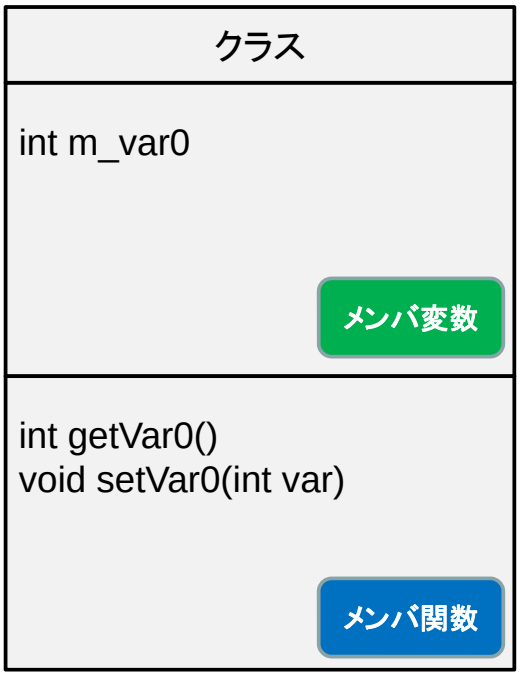
- オブジェクト指向とは
  - カプセル化（≒モジュール化）
    - 処理の詳細とデータの隠蔽
  - インヘリタンス
    - 機能の継承や拡張
  - ポリモーフィズム
    - 多態性と共通化
  - ダイナミックバインディング
    - 型の違いによる処理の差別化

とりあえず
オブジェクト指向＝カプセル化
と覚えておけば十分

クラスの概念

クラス ＝ データ ＋ 処理

- メンバ変数
  - インスタンスと同時に生成される変数
  - オブジェクトの状態を保持
  - 通常、オブジェクトの外からは見えないようにする（隠蔽）
- メンバ関数
  - オブジェクトに作用を及ぼす
    - メンバ変数の値を変える
    - オブジェクトに処理をさせて結果を得る



なぜ変数を隠蔽するのか

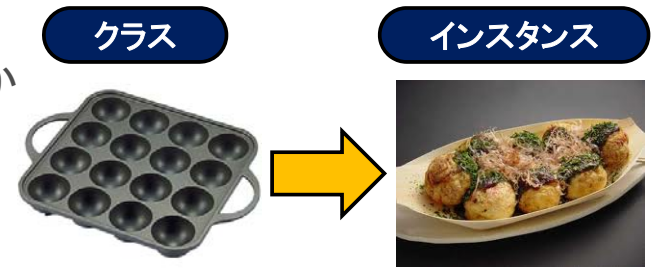
- 実装の隠蔽
- 実装と振る舞いの分離
- 依存性の排除
- 隠蔽レベルの制御(C++の例)
  - public
  - protected
  - private

2つの setVar の例

```
void setVar0(int var)
{
  m_var0 = var;
}

void setVar0(int var)
{
  if (var > 100) { m_var0 = 100; }
  else if (var < 0) {m_var0 = 0; }
  else { m_var0 = var; }
}
```


クラスとインスタンス

- クラス
  - オブジェクトを作るための型
  - メンバ変数、メンバ関数から構成される
- インスタンス
  - オブジェクトとも呼ぶ
    - (厳密には両者は違うが…)
  - クラスを具現化したもの
  - クラス:たこ焼きの型
  - インスタンス:たこ焼き



実習（2）

クラスを使ったHello World

```
#include <iostream>
#include <string> // std::stringを使うのに必要

class HelloWorld
{
public:
  HelloWorld(std::string str = "World")
   : m_str(str)
  {
  }
  void printHello()
 {
    std::cout << "Hello, " << m_str << std::endl;
 }
private:
  std::string m_str;
};
```

```
int main()
{
  HelloWorld hello0;
  HelloWorld hello1("C++ world");

  hello0.printHello();
  hello1.printHello();

  return 0;
}
```


AIST RT MIDDLEWARE