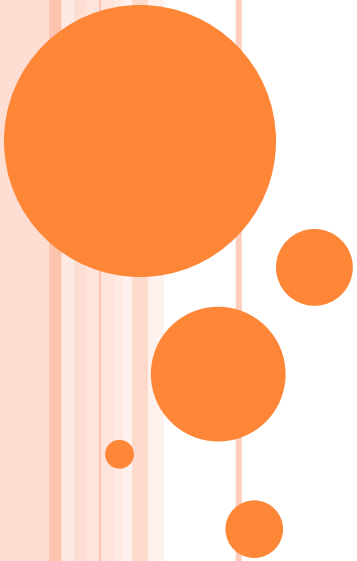


JAVA入門

今回の内容

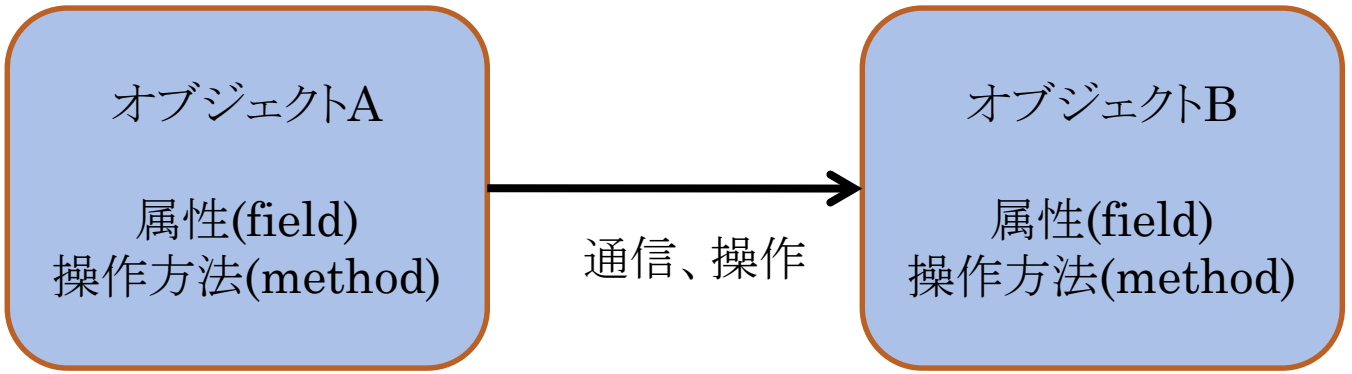
- グラフとオブジェクト指向プログラミング
  - Javaを使う理由
- Javaの基本
  - Javaのライブラリ
  - 開発・実行
- クラスの再利用
  - クラス継承
  - 抽象クラス
- 開発の要点

グラフを記述するには

- 頂点(Vertex)と弧(Arc)、その間の関係
- 素直にデータ構造として表現したい
  - グラフは、頂点と弧の集合
  - 弧から始点と終点を得る
  - 頂点から、その頂点を始点とする弧の集合を得る
- 頂点と弧をモノ (object) として捉える
  - モノを中心にプログラムを考える枠組みが欲しい
  - オブジェクト指向プログラミング

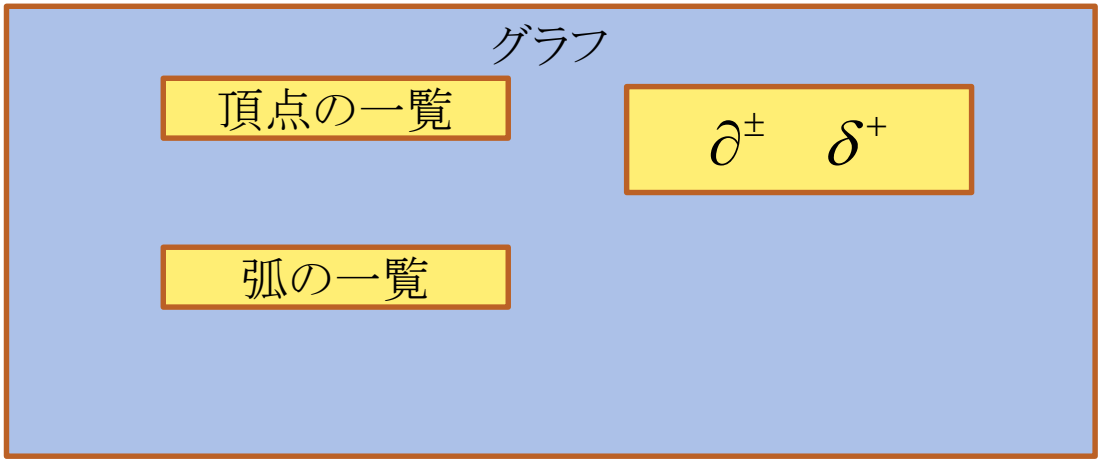
オブジェクト指向(Object Oriented)

- もの(object)の操作・動作を中心に考える
  - 操作や動作を日本語で考え、出てくる名詞に注目する
- オブジェクトの構成
  - 属性(field):データなど
  - 操作(method)



グラフをオブジェクト指向プログラミングで考える

- グラフの構造を表すデータ構造
  - グラフ、頂点、弧
- 階層的データ
  - グラフの要素としての頂点と弧
  - 頂点に接続している弧のリスト
  - 弧の両端の頂点



- 各データごとの操作
  - 弧に値を設定する
  - 探索問題：頂点や弧に印を付ける
- データのカプセル化
  - グラフとしての整合性を維持
  - 頂点や弧に属性等を追加
- 型の継承と拡張
  - 弧に「流れ」の属性を付けて拡張
- グラフの可視化

様々なOOP言語

- Smalltalk80
  - Xerox, Palo Alto研究所
- C++
  - B. Stroustrup
  - C にOOPを導入
- Java
  - Sun Microsystems (Oracle)
- Ruby
  - まつもとひろゆき
  - スクリプト言語

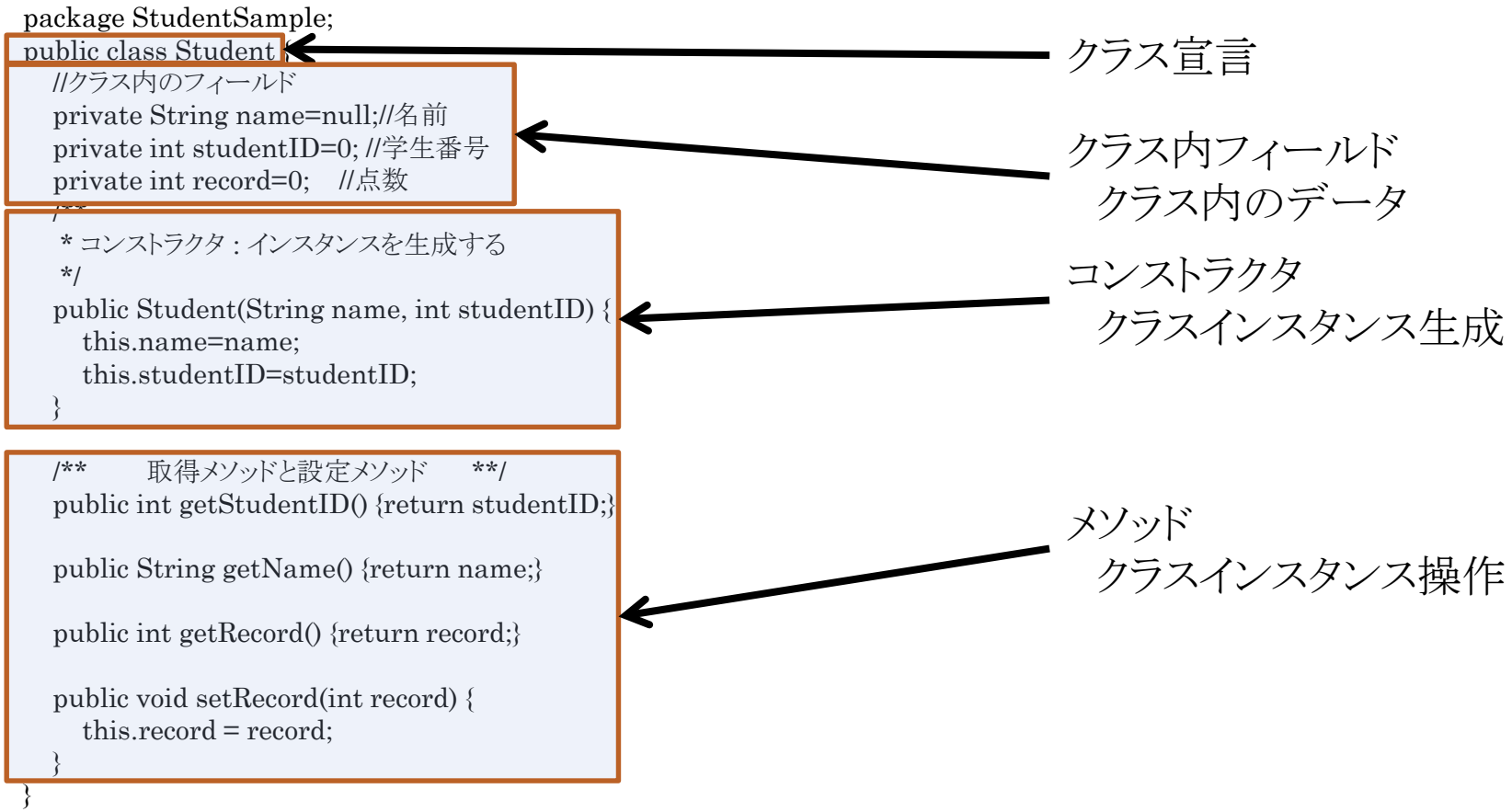
C++ではなくJAVAを使う理由

- 豊富なユーティリティー
  - java.util.ArrayListなど
- 使い易い開発環境(IDE)
  - NetBeans、Eclipse
- 多数のOSで使える
  - Windows、Linux、Solaris
  - OSに依存しない実行形式
- GUI開発が容易
  - IDEを使うと簡単

Javaの基本

- 全てがクラス
- 開始点となるクラスが必要
  - public static void main(String[] args)メソッドから始まる
  - mainは主となるクラスを起動するだけ
- コンストラクタメソッド
  - クラスと同じ名前のメソッド
- デストラクタは無い
  - 自動ガベージコレクション
- 一つのクラスで一つのファイルが基本
  - ファイル名はクラス名と同じ
- ヘッダファイルが無い
  - ライブラリはimport文を使う
- C/C++のポインタは無い
  - 原始型は値代入
  - クラスオブジェクトは参照

- 文法はだいたいC++と同じ
- 原始型はint、double、char、booleanなど
- 原始型に対応したクラスがある
  - Integer、Double、Character、Booleanなど
- 文字列Stringや原始型の配列はクラス
- ポインタが無い
- デストラクタは書かない
  - 不要なオブジェクトは自動で削除される
- クラスは階層化され、パッケージになっている


package StudentSample;
public class Student {
//クラス内のフィールド
private String name=null;//名前
private int studentID=0; //学生番号
private int record=0;　//点数
/**
* コンストラクタ：インスタンスを生成する
*/
public Student(String name, int studentID) {
this.name=name;
this.studentID=studentID;
}
/**　　取得メソッドと設定メソッド　　**/
public int getStudentID() {return studentID;}
public String getName() {return name;}
public int getRecord() {return record;}
public void setRecord(int record) {
this.record = record;
}
}
クラス宣言
クラス内フィールド
クラス内のデータ
コンストラクタ
クラスインスタンス生成
メソッド
クラスインスタンス操作

便利なライブラリ

- オンラインマニュアル
  - http://docs.oracle.com/javase/8/docs/api/
- 基本的なクラス：java.lang
- 入出力：java.io
- コレクション(リストなど)：java.util
- 基本GUI：java.awt
- 拡張GUIセットSwing：javax.swing

開発環境

- http://www.oracle.com/technetwork/java/javase/downloads/index.html
- NetBeans
  - http://www.netbeans.org/index_ja.html
- プロジェクト管理
- メソッド名の補完
- パーツを使ったGUI構築
- CVS等を使ったバージョン管理

プログラム開発の手順

- 作業ディレクトリを決める
  - デフォルトでは~/Documents/NetBeansProjects
- NetBeansを起動
- 「ファイル」→「新規プロジェクト」
- プロジェクトウィンドウ内で
- プロジェクト名→「ソースパッケージ」→「デフォルトパッケージ」で右ボタン「新規」

構築と実行

- GUIの無い主クラス
  - 「Java 主クラス」
- GUIのある主クラス
  - 「JFrameフォーム」
- テンプレートを上手に使う

- プロジェクトウィンドウ内で
  - プロジェクト名→「プロジェクトを構築」
- プロジェクトウィンドウ内で
  - プロジェクト名→「プロジェクトを実行」
  - 主クラス名→「ファイルを実行」
- デフォルトでは、ファイルを保存すると、コンパイルする

OOPと開発効率

- OOPはプログラム開発効率を改善する
- カプセル化
  - クラス内部の構造を隠す
  - 変更をクラス内に止め、他に影響を与えない
- クラスの継承・再利用
  - 機能や属性を既存のクラスに追加する
- 抽象クラス
  - 機能や属性の似たクラスをグループ化する

プログラム開発の要点

- 開発・保守コストを下げる
- クラスの再利用
  - ルーチン化したコードを再利用
  - 他の人のノウハウを借用
- 分かりやすい構成
  - 自分にも他人にもわかるように
  - 修正箇所の限定
  - 修正の影響範囲を明確化

プログラム開発の要点2

- アルゴリズムをデータの詳細と切り離す
  - ソートのアルゴリズムは、ソートされるデータの詳細とは関係ない
  - スレッドプログラムは、各スレッド内で何をしているかと関係ない
- 問題をオブジェクトの運動として捉える
  - 小さなオブジェクトへ分割
  - 小さなオブジェクトならば、その役割が明確になる