Programming II

第14回 Objectクラス

塩澤秀和

14.1 Objectクラス

- Objectクラス（p.530）
  - Javaのすべてのクラスの親（先祖）となるクラス
  - クラス定義にextendsがなくても，暗黙に継承される
  - 例）class Empty {}

    ↓ 実は

    class Empty extends Object {}

- Objectクラスの役割
  - Javaのクラスが必ず持つメソッドが定義されている
    - toString: 文字列表現，equalsとhashCode: 等しいか判定
  - 多態性と組み合わせて「任意のクラス」を表す
    - どんなクラスのインスタンスも許容する変数，配列，引数
  - インタフェースもObjectを継承しているように振る舞う

14.2 toStringメソッド

- 文字列表現の生成（p.535）
  - インスタンスの文字列としての表現を生成する
  - 文字列との足し算（+）やprint系のメソッドで利用される
  - 通常は，属性の値を表示するようにオーバーライドする
  - 例）

```
public class Person {
    private String firstName, lastName;

    // いろいろ省略

    @Override
    public String toString() {
        return firstName + " " + lastName;
    }
}
```

文字列（String）を返す

14.3 equalsメソッド

- 意味的に「等しい」か判定する
  - 基本的に，教科書p.538と同じようにまねして定義する

```
public boolean equals(Object obj) {
    if (this == obj) return true;  // 同一のインスタンスなら真
    if (obj instanceof クラス名) {  // 同種のクラスなら中身で判定
        クラス名 a = (クラス名) obj;
        if (thisとaの中身が意味的に等しいか比較する) {
            return true;
        }
    }
    return false;
}
```

引数の型に注意！

1つのif文では書けずに
複雑になることもある

  - 実は，hashCodeメソッドも定義すべきだが，省略…

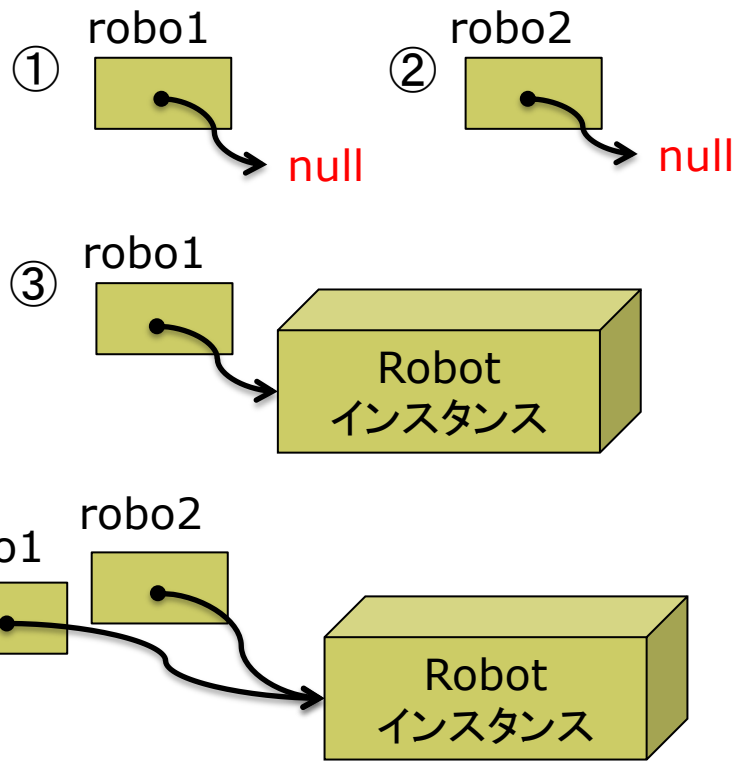
14.4 == と equals の違い

- Javaで「等しい」かどうかの比較（p.537）
  - 基本型か参照型か，==かequalsかで意味が異なる
  - 教科書では，「等値」vs「等価」という表現（非公式用語）

比較方法	a == b	a.equals(b)
基本型 （intやdouble）	• 値（中身）が等しい	
参照型 （クラスや配列）	• 同一性（等値） • equality / identity • 参照しているインスタンス（実体）が，同一のものであるか判定する • a==b ならば，常に，a.equals(b) も成り立つ	• 同値性（等価） • equivalence • オーバーライドされた定義に基づいて，内容が意味的に等しいか判定する • 配列に対しては使わないこと（期待通りに動作しない）

4.4 クラス型の変数とインスタンス

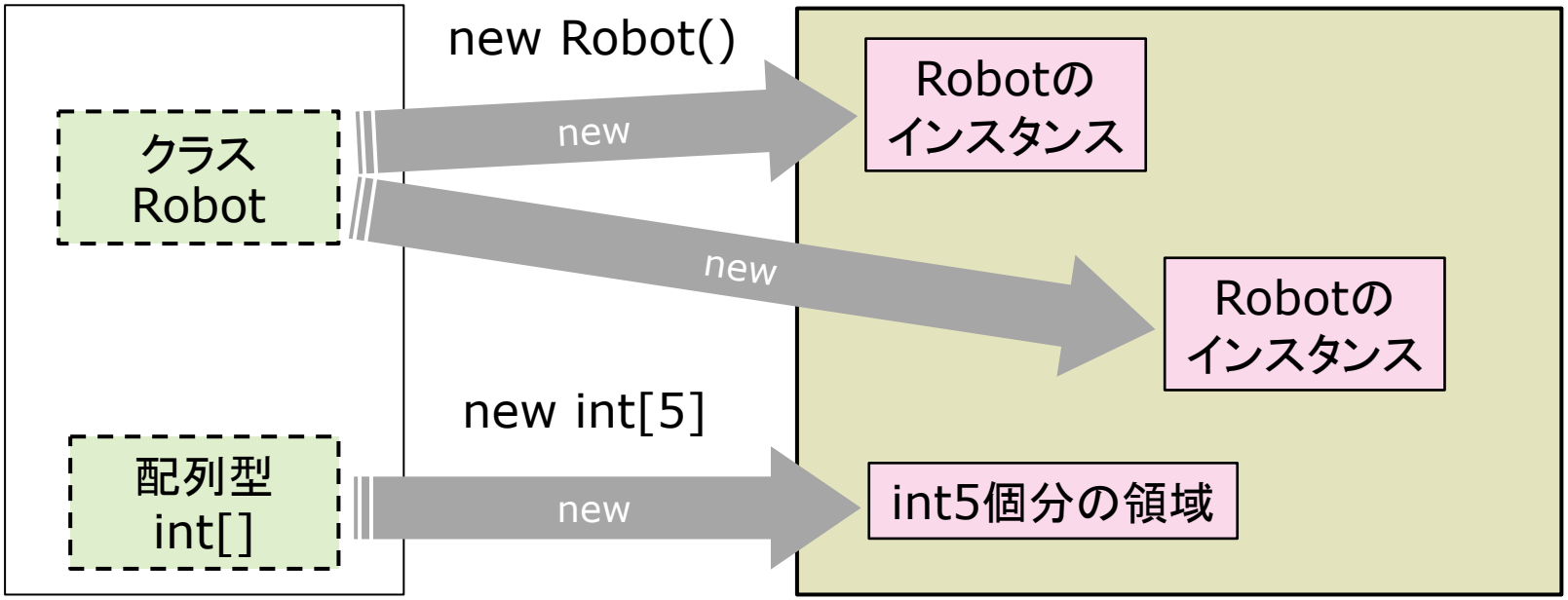
- クラスは参照型(p.334)
  - 実体(インスタンス)を代入するまで，変数は空(null)
  - newで作った実体を代入する
    - ① Robot robo1;
    - ② Robot robo2;
    - ③ robo1 = new Robot();
    - ④ robo2 = robo1;



- 復習：配列も同じ参照型
  - 以下を同様に図示せよ
    - ① int [] data1;
    - ② data1 = new int [5];
    - ③ int [] data2 = data1;

4.5 インスタンスの生成

- new演算子の働き（p.334）
  - コンピュータの中のメモリに領域を確保している
  - 1回のnewで1個のインスタンスができる 👈 これ重要！



演習

- 演習課題 1.
  - 座標を表すCoordinateクラスのインスタンスを2個作り，それぞれのxとyの値をキーボードから読み込み，それら2つが等しいかequalsメソッドで判定するようにせよ
  - 最終的なソースコードと実行結果を提出

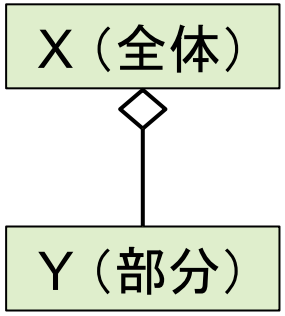
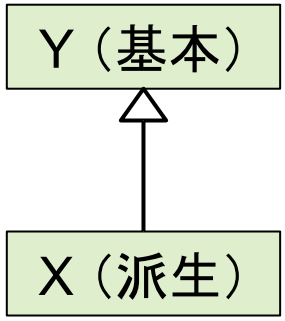
14.5 オブジェクト指向開発

- オブジェクト指向分析設計
  - ソフトウェアシステムの要件を明確化し，それを実現するために必要な部品（オブジェクト）とその機能を分析する
  - さらに，オブジェクト同士の関係（has-a，is-a等），動作のタイミング，情報のやり取りなどを分析・設計する
  - その際，「デザインパターン」などの設計技法を利用する
  - オブジェクト（クラス）の関係を，UMLなどで記述する

- オブジェクト指向プログラミング（OOP）
  - オブジェクト指向用の言語・文法でプログラムを記述する
  - 全体の設計にオブジェクト指向を用いない場合でも，外部サービスの利用などでOOPが必要になることが多い

14.6 HAS-A vs IS-A

- 復習しておくこと

	HAS-A	IS-A
意味	他のクラスを属性として持つ X has a Y	他のクラスを継承する X is a Y
クラス図		

Java	class X {
	Y y;
	// …
	}
		class X extends Y {
		// …
		}

2.8 Javaの標準ライブラリ

- Java API（Application Program Interface）（p.255）
  - Javaで標準クラスとして提供されている機能
  - 詳細は，マニュアル（APIリファレンス）を参照

パッケージ	機能	クラスの例
java.lang	基本機能（暗黙にimport）	String, Math, Object, Thread
java.util	さまざまな便利な機能	Scanner, Date, ArrayList
java.io	ファイル等の入出力関連	File, FileReader, InputStream
java.net	ネットワーク関連	Socket, HttpURLConnection
java.sql	データベースへのアクセス	DriverManager
java.awt javax.swing	GUI，ウィンドウ表示	Color, Font, JFrame, JButton

第14回 まとめ

- Objectクラス
- toStringメソッド
- equalsメソッド
- == と equals の違い
- オブジェクト指向開発
- has-a と is-a の区別