（2008-05-20 版）

Java によるプログラミング入門 (7)

1　Lesson 5 オブジェクトとクラス

いよいよ，“オブジェクトを用いて扱いたい対象のモデルを構成する” ための流れを学びます.

1.1　タスク 5

以下のような想定で給水管から直列につながった 3 台のタンクに水がたまる様子をシミュレーションします.

- 給水管からは一定量の水が供給されるとします.
- 各タンクの排水口から水が次のタンクに送られますが，各タンクの排水口から流出する水の流量はそのタンクの水位に比例するものとします．すなわち

$$\text{タンクの単位時間排水量} = \text{排水係数} \times \text{タンクの水位} \tag{1}$$

- 給水管と各タンクのパラメータは次の表のように与えられます.

給水管からの流入量	$1\ m^3/s$		
タンク	タンク 1	タンク 2	タンク 3
底面積	$10\ m^2$	$5\ m^2$	$4\ m^2$
初期水位	$1\ m$	$3\ m$	$1\ m$
排水係数	$0.5\ m^3/ms$	$0.3\ m^3/ms$	$0.6\ m^3/ms$

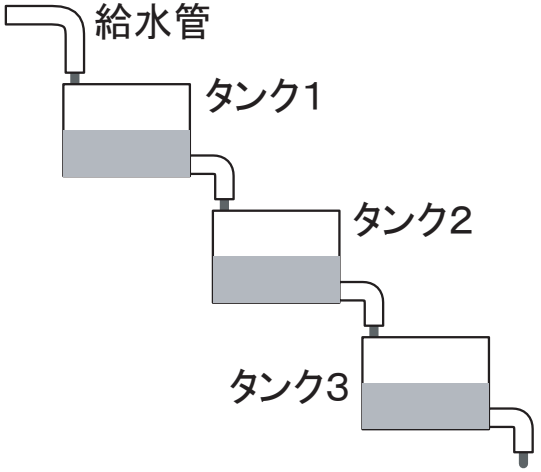


図 1　3 タンクの問題

1.2 モデル・アルゴリズム・オブジェクト

複雑な対象を考えてシミュレーションするためには，それを何らかの**「モデル」**として表現し，コンピュータ上に構成する必要があります．

これまで java について学んできたことは，行いたいことを**一連の操作手順**で表すことでした．どちらかというと操作の時間的な展開を考えてきたといえます．そして「一連の手順」は**「アルゴリズム」**と呼ばれます．

これに対して，複雑な対象を効果的にモデルとして表現する手段が**「オブジェクト」**です．どちらかというと対象の空間的な広がりを効果的に表すためのものです．

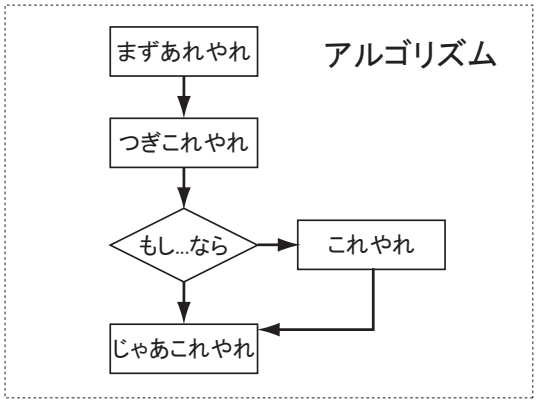


図 2

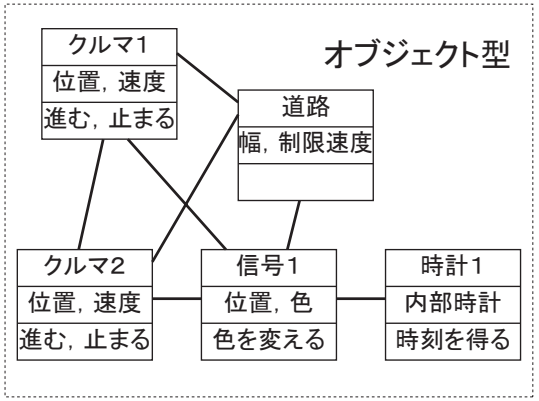


図 3

オブジェクト指向の設計・プログラミングは次のような手順で行います．

1. **【モデルの記述】** やりたいことを詳しく書き出します．この例ではタンクの問題を差分方程式（漸化式）で数学のモデルとして記述します．
2. **【モデルの整理と切り出し】** 対象を表すにはどのような「もの」と「操作」が必要かという視点で問題を整理してゆきます．
3. **【オブジェクトの記述と単体のプログラミング】** このように整理した概念からオブジェクトとして対象を記述して，プログラムを構成していきます．
4. **【組み上げ】** やりたいことに合わせてシステムを組み上げます．

1.3 （1）モデルの記述 ・・・ タンクの数学モデル

このシステムはタンク $i = 1, 2, 3$ の貯水量を $V_i(t)$ とすると次の微分方程式[1] で表されます（別にこの例が分からなくても問題ないです）.

このために使う変数を定義しましょう：t は時間，$h_i(t)$ はタンク i の水位です．また，F, k_i, S_i はそれぞれ給水管からの流入量，タンク i の排水係数と底面積とします．このとき各タンクの動作は次のように表されます：

$$\frac{dV_1(t)}{dt} = F - k_1 h_1(t) \tag{2}$$

$$\frac{dV_2(t)}{dt} = k_1 h_1(t) - k_2 h_2(t) \tag{3}$$

$$\frac{dV_3(t)}{dt} = k_2 h_2(t) - k_3 h_3(t) \tag{4}$$

$$h_1(t) = \frac{V_1(t)}{S_1} \tag{5}$$

$$h_2(t) = \frac{V_2(t)}{S_2} \tag{6}$$

$$h_3(t) = \frac{V_3(t)}{S_3} \tag{7}$$

なお時刻 $t = 0$ での水位 $h_i(0)$ が初期値として与えられます（時刻 0 での貯水量 $V_i(0)$ は水位から決まります）.

[1] とりあえず，方程式を解くということは**考えない**で下さい．解くことはコンピュータの仕事です．微分方程式が苦手な人は次の差分方程式から考え始めても構いません.

1.4 （1）モデルの記述 ・・・ タンクの数学モデル

これを時間幅 Δt(2 文字ですがこれで 1 つの変数です） で差分近似すると以下の式を得ます．[2] **これは漸化式なので，時間を追っての計算は特に難しくない**はずです：

$$V_1(t+\Delta t) = V_1(t) + \Delta t(F - k_1 h_1(t)) \tag{8}$$
$$V_2(t+\Delta t) = V_2(t) + \Delta t(k_1 h_1 - k_2 h_2(t)) \tag{9}$$
$$V_3(t+\Delta t) = V_3(t) + \Delta t(k_2 h_2 - k_3 h_3(t)) \tag{10}$$
$$h_1(t) = \frac{V_1(t)}{S_1} \tag{11}$$
$$h_2(t) = \frac{V_2(t)}{S_2} \tag{12}$$
$$h_3(t) = \frac{V_3(t)}{S_3} \tag{13}$$

ただし，例えばあるタンク流入量より流出量のほうが多い場合に，大きな Δt を用いると状況によってはタンクの水位が負になってしまうことがあります．これは差分近似を行ったことにより生じる不都合ですが，以下のように簡単なルールを加えておきましょう．

もし $V_i(t+\Delta t)$ が負になったら，これを 0 に設定しなおす．

[2] これまでは時間幅を 1 秒としてきましたが，ここではタンクからの流出量がタンクの水位に依存して変わることをよい近似で計算するために時間幅をパラメータ Δt としておきます．

1.5 （2）モデルの整理と切り出し

では次にこのモデルの分析をして行きます．まずモデルに現れる概念を書き出し，それが**「名詞」**か**「動詞」**かに着目しながら，整理していきます．概念は・・・

時刻 (t), 時間の進み幅 (Δt), 初期時刻，時刻を更新する，給水管，給水管からの給水量 (F), タンク $(i = 1, 2, 3)$, タンクへの流入元（給水管とか前のタンク），タンクへの流入量，タンクの底面積，タンクの貯水量，タンクの初期水位，タンクの排水係数，タンクからの流出量，タンクの貯水量を更新する，タンクの水位を計算する

これを少しグルーピングしてみると・・・

時間に関わるもの ：

名詞: 時刻，時間の刻み幅，初期時刻, **動詞**: 時刻を更新する

給水管に関わるもの ：

名詞: 給水管，給水量

タンク 1,2,3 **に関わるもの** ：

名詞: タンク，底面積，貯水量，初期水位，排水係数，流入量，流出量

動詞: 貯水量を更新する，水位を計算する

タンクの接続に関わるもの（全体のシステム） ：

名詞 タンクの流入元

ここで下線を引いたものは単純に一つの「数値」として表現できるものです．

1.6 （3）オブジェクトの記述と単体のプログラミング

上記のように整理したものを次にオブジェクトとして構成して行きます．すなわち，**クラスを書き，フィールドやメソッドを定めます**．その際の方針は

- 先の分析で「下線のつかない名詞」（数値でも機能でもないもの）が**オブジェクト**の候補です．

  - 給水管，タンク１２３など

- 動詞はそれと関係するオブジェクトのメソッドの候補です．

  - タンク１における，「水位を計算する」など

- そのオブジェクトの属性となる名詞（特に下線を引いたもの）で，永続性のあるものは**フィールド**の候補です．

  - タンクの貯水量はフィールドになります．流出量はなりません

そしてそのクラスに必要なフィールド・メソッドについて，フィールドの型は何か，定数か．メソッドは値を返すか，引数は何か．などを考えます．ここまでできれば，一行もコードを書いていなかったとしても半分は終わったようなものです．

続いて，これらをプログラムとして記述していきます．

- 他のオブジェクトを必要としない単純なものから構成して行きます．
- オブジェクトを構成したら，その中に main メソッドを書いて [3] テストするようにしましょう．

[3] Java では各クラスが main メソッドを持つことができ，これを実行できます．各クラスにそれをテストする main メソッドを書けば他のクラスに依存しないクラスから順にテスト可能です．

上記の方針から以下，給水管，時間に関わるもの，タンク１，２，３，全体のシステムの順にプログラムを作って行きます．以下の実装例の設計方針を簡単に説明します：

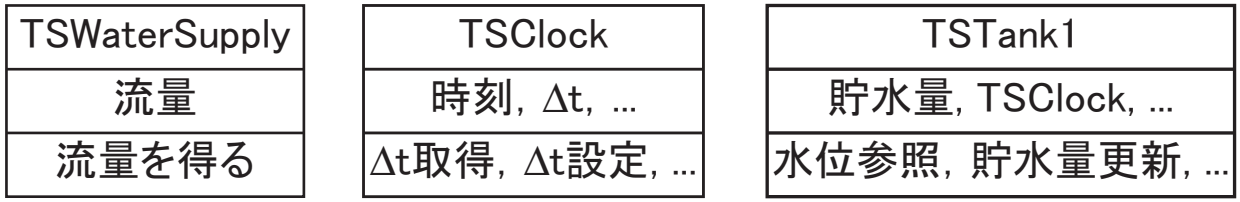
TSWaterSupply　:給水管の流量を（定数）フィールドとして持ち，これを参照するメソッドを持ちます．

TSClock　: 時間を管理するクラスです．初期時刻, 時間ステップ, 時刻などをフィールドに，これらを設定したり，参照したりするメソッドを持ちます．また時刻を更新するメソッドを持ちます．

TSTank1　:タンク 1 に関するクラスです．フィールドとして底面積，初期水位，排出係数，貯水量，流入量と時間を管理するクラス TSClock のオブジェクトを持ちます．コンストラクタとして，TSClock を引数から得てこれを設定し，水位を初期化するものを持ちます．メソッドとしては水位の参照，流入量の設定と参照，排出係数の参照，貯水量の更新などを持ちます．

TSTank2　: TSTank1 に準じたタンク 2 用のクラスです．

TSTank3　: TSTank1 に準じたタンク 3 用のクラスです．



1.7 実装例

■TSWaterSupply.java — **給水管**

```
 1: public class TSWaterSupply {
 2:   private final double flowRate = 1.0; //m**3/s
 3:   public double getFlowRate() {
 4:     return flowRate;
 5:   }
 6:   public static void main(String args[]) {
 7:       TSWaterSupply tsWaterSupply = new TSWaterSupply();
 8:     System.out.println("Flow Rate of Water Supplu = " + tsWaterSupply.getFlowRate());
 9:   }
10: }
```

※今日の資料に載っているプログラムは，すべて

http://130.54.13.233/lecture/java/lesson567.zip にあります.

ダウンロードして，これらのファイルを M:¥java に入れてください.

■TSTank1.java — **タンク** 1

```
 1: public class TSTank1 {
 2:   private final double TANK_AREA = 10; //m**2
 3:   private final double INITIAL_TANK_LEVEL = 1; //m
 4:   private final double DRAINAGE_COEFFICIENT = 0.5; //m**3/s.m
 5:   private double storedVolume;
 6:   private double inFlow;
 7:   private TSClock tsClock;
 8:
 9:   public TSTank1(TSClock tsClock) {
10:       this.tsClock = tsClock;
11:       storedVolume = INITIAL_TANK_LEVEL * TANK_AREA;
12:   }
13:
14:   public double getTankLevel() {
15:       return storedVolume/TANK_AREA;
16:   }
17:
18:   public void setInFlow(double inFlow) {
19:       this.inFlow = inFlow;
20:   }
21:
22:   public double getInFlow() {
23:       return inFlow;
24:   }
25:
```

```
26:   public double getDrainageRate() {
27:       return DRAINAGE_COEFFICIENT*getTankLevel();
28:   }
29:
30:   public void update() {
31:       storedVolume += tsClock.getTimeStep()*(inFlow - getDrainageRate());
32:       if (storedVolume < 0) {
33:           storedVolume = 0.0;
34:       }
35:   }
36:
37:   public static void main(String args[]) {
38:       TSClock tsClock = new TSClock();
39:       tsClock.setTimeStep(0.1);
40:       TSTank1 tsTank1 = new TSTank1(tsClock);
41:       System.out.println("time,tank level");
42:       for (int i=0;i<100;i++) {
43:           tsTank1.setInFlow(1.0);
44:           System.out.println(tsClock.getTime() + "," + tsTank1.getTankLevel());
45:           tsClock.update();
46:           tsTank1.update();
47:       }
48:   }
49: }
```

■TSTank2.java — **タンク** 2

```
 1: public class TSTank2 {
 2:   private final double TANK_AREA = 5; //m**2
 3:   private final double INITIAL_TANK_LEVEL = 3; //m
 4:   private final double DRAINAGE_COEFFICIENT = 0.3; //m**3/s.m
 5:   private double storedVolume;
 6:   private double inFlow;
 7:   private TSClock tsClock;
 8:
 9:   public TSTank2(TSClock tsClock) {
10:       this.tsClock = tsClock;
11:       storedVolume = INITIAL_TANK_LEVEL * TANK_AREA;
12:   }
13:
14:   public double getTankLevel() {
15:       return storedVolume/TANK_AREA;
16:   }
17:
18:   public void setInFlow(double inFlow) {
19:       this.inFlow = inFlow;
20:   }
21:
22:   public double getInFlow() {
23:       return inFlow;
24:   }
25:
```

```
26:   public double getDrainageRate() {
27:       return DRAINAGE_COEFFICIENT*getTankLevel();
28:   }
29:
30:   public void update() {
31:       storedVolume += tsClock.getTimeStep()*(inFlow - getDrainageRate());
32:       if (storedVolume < 0) {
33:           storedVolume = 0.0;
34:       }
35:   }
36:
37:   public static void main(String args[]) {
38:       TSClock tsClock = new TSClock();
39:       tsClock.setTimeStep(0.1);
40:       TSTank2 tsTank2 = new TSTank2(tsClock);
41:       System.out.println("time,tank level");
42:       for (int i=0;i<100;i++) {
43:           tsTank2.setInFlow(1.0);
44:           System.out.println(tsClock.getTime() + "," + tsTank2.getTankLevel());
45:           tsClock.update();
46:           tsTank2.update();
47:       }
48:   }
49: }
```

■TSTank3.java — **タンク** 3

```
 1: public class TSTank3 {
 2:   private final double TANK_AREA = 4; //m**2
 3:   private final double INITIAL_TANK_LEVEL = 1; //m
 4:   private final double DRAINAGE_COEFFICIENT = 0.6; //m**3/s.m
 5:   private double storedVolume;
 6:   private double inFlow;
 7:   private TSClock tsClock;
 8:
 9:   public TSTank3(TSClock tsClock) {
10:       this.tsClock = tsClock;
11:       storedVolume = INITIAL_TANK_LEVEL * TANK_AREA;
12:   }
13:
14:   public double getTankLevel() {
15:       return storedVolume/TANK_AREA;
16:   }
17:
18:   public void setInFlow(double inFlow) {
19:       this.inFlow = inFlow;
20:   }
21:
22:   public double getInFlow() {
23:       return inFlow;
24:   }
25:
```

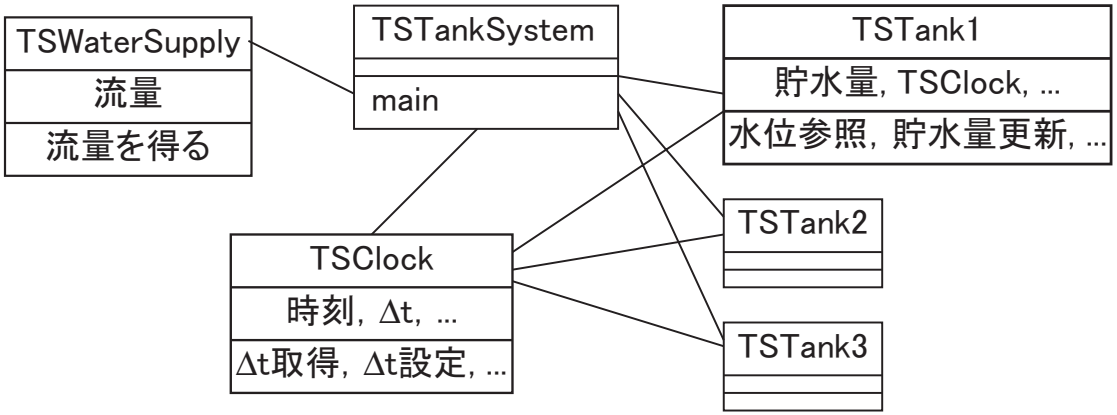
```
26:   public double getDrainageRate() {
27:       return DRAINAGE_COEFFICIENT*getTankLevel();
28:   }
29:
30:   public void update() {
31:       storedVolume += tsClock.getTimeStep()*(inFlow - getDrainageRate());
32:       if (storedVolume < 0) {
33:           storedVolume = 0.0;
34:       }
35:   }
36:
37:   public static void main(String args[]) {
38:       TSClock tsClock = new TSClock();
39:       tsClock.setTimeStep(0.1);
40:       TSTank3 tsTank3 = new TSTank3(tsClock);
41:       System.out.println("time,tank level");
42:       for (int i=0;i<100;i++) {
43:           tsTank3.setInFlow(1.0);
44:           System.out.println(tsClock.getTime() + "," + tsTank3.getTankLevel());
45:           tsClock.update();
46:           tsTank3.update();
47:       }
48:   }
49: }
```

■TSClock.java — **時間**

```
 1: public class TSClock {
 2:   private final double INITIAL_TIME = 0.0; //s
 3:   private double timeStep;
 4:   private double time;
 5:   public TSClock() {
 6:       time = INITIAL_TIME;
 7:   }
 8:   public void setTimeStep(double timeStep) {
 9:       this.timeStep = timeStep;
10:   }
11:   public double getTimeStep() {
12:       return timeStep;
13:   }
14:   public double getTime() {
15:       return time;
16:   }
17:   public void update() {
18:       time += timeStep;
19:   }
20:   public static void main(String args[]) {
21:       TSClock tsClock = new TSClock();
22:       tsClock.setTimeStep(0.1);
23:       for (int i=0;i<10;i++) {
24:           System.out.println("Iteration = " + i + " Time = " + tsClock.getTime());
25:           tsClock.update();
26:       }
27:   }
28: }
```

1.8 （4）システム全体を組み上げる

ここまで準備ができればあとはシステム全体を組み上げるだけです．TSTankSystem.java というクラス内の main メソッドでシステムを組み上げていくことにします．必要なクラスのインスタンスを作り，メソッドを呼ぶことでタンクのシミュレーションが行えます．どのタンクがどれに繋がっているのかが書かれているのは，このメインメソッドです．



このように，登場するクラス，そのメソッド・フィールド，さらに関係を書いた図を**クラス図**と呼びます．全てのクラスに関係がある（相手を知っている）必要はない，というのがポイントです．

■TSTankSystem.java — **システム全体**

```
 1: public class TSTankSystem {
 2:     public static void main(String args[]) {
 3:         TSClock tsClock = new TSClock();
 4:         tsClock.setTimeStep(0.1);
 5:         TSWaterSupply tsWaterSupply = new TSWaterSupply();
 6:         TSTank1 tsTank1 = new TSTank1(tsClock);
 7:         TSTank2 tsTank2 = new TSTank2(tsClock);
 8:         TSTank3 tsTank3 = new TSTank3(tsClock);
 9:         System.out.println("time,tank level 1, tank level 2, tank level3");
10:         for (int i=0; i<300; i++) {
11:             tsTank1.setInFlow(tsWaterSupply.getFlowRate());
12:             tsTank2.setInFlow(tsTank1.getDrainageRate());
13:             tsTank3.setInFlow(tsTank2.getDrainageRate());
14:             System.out.println(tsClock.getTime() + "," + tsTank1.getTankLevel()
15:               + "," + tsTank2.getTankLevel() + "," + tsTank3.getTankLevel());
16:             tsClock.update();
17:             tsTank1.update();
18:             tsTank2.update();
19:             tsTank3.update();
20:         }
21:     }
22: }
```

実際に，TSTankSystem.java をコンパイルし，実行してみてください．このとき，必要な他の java ファイルも自動的にコンパイルされ，TSClock.class などが生成されていることを確認しましょう．

実行結果をファイルに保存し，表計算ソフトでグラフ化した各タンクの水位の変化を次の図に示します．

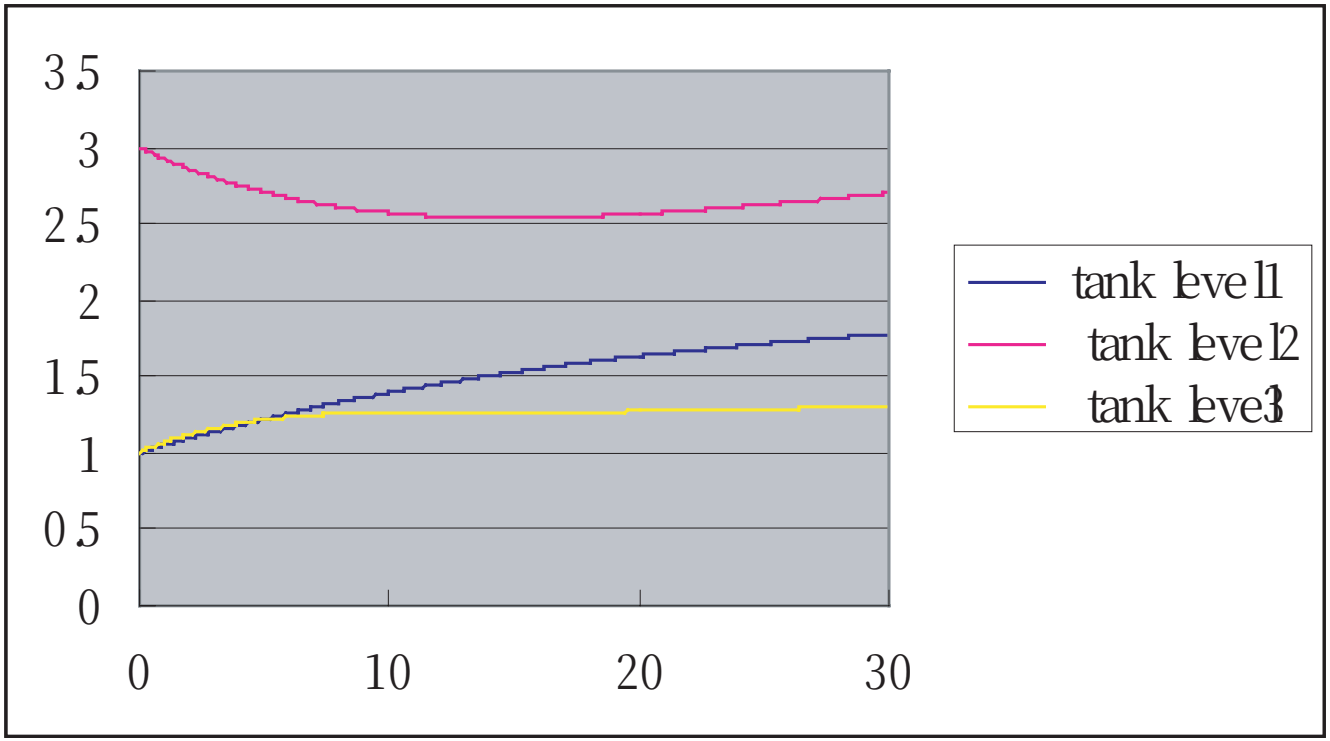


図 4　3 タンク問題のシミュレーション結果

1.9　メソッドは誰のものか

実際に，何らかのテーマ・ストーリー・問題に対して，オブジェクトの記述をしていこうとすると，２つ以上のオブジェクトに関連するメソッドを，どちらのものとして考えたら良いか，わからなくなることがあります．

運転手Ａさんが車Ｂに乗っており，ブレーキをかけて止めました

こんな事象を，オブジェクトとして記述してみたいとします．オブジェクトとして運転手Ａ，車Ｂを切り出すのは良いとして，以下の２通りが考えられます．

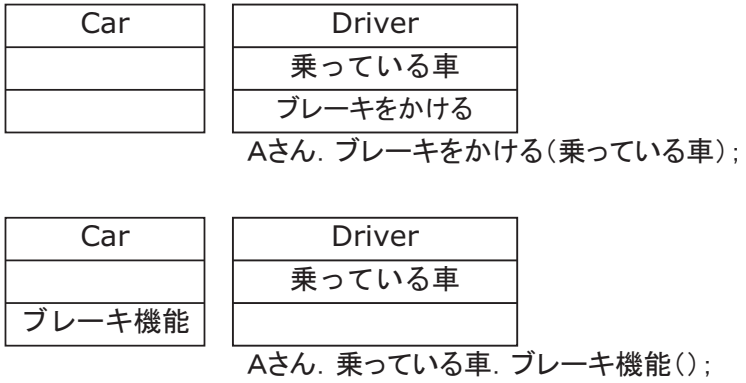


図５　２パターンのオブジェクト記述

どちらも正しいように見えますが，こういう場合「その機能を使うのは誰」「実際に仕事をするのは誰」ということを考え，実際に仕事をする側のメソッドとして定義します．なぜなら，処理内容は実際に仕事をする側でなければ分からないからです．使う側は，使い方だけ知っていれば良いのです．

1.10　演習：オブジェクトの記述をしてみよう

もう一つ，簡単な例を挙げておきます．この文章について，オブジェクトを記述してみてください．

Aさんは，電子レンジで肉じゃがを温め，食べました

ポイントは，「温める」「食べる」が誰のメソッドであるか，というところです．

1.11　演習：オブジェクトの記述をしてみよう

もっと難しいシナリオです．(a) 7 ページを参考に モデルの整理と分析を行い，動詞と名詞をテキストファイルに書き出してください．(b) そしてその中から 8 ページを参考にオブジェクトの候補を探し，そのオブジェクトのフィールドとメソッドをテキストファイルに書き出してください．(c) 最後に，9, 18 ページを参考にオブジェクトの関係を A4 紙に（大きめに）書いてみてください．

Ａさんが，道ばたで倒れている病人を見つけました．Ａさんは，救急センターの 119 番機能を利用しました．Ａさんは座標を聞かれました．救急センターは救急車Ｂ，Ｃというリストを持っています．救急車Ｂ，Ｃの座標確認機能を利用したところ救急車Ｃのほうが近いことが分かりました．救急センターは救急病院Ｄ，Ｅというリストを持っています．救急センターは病院Ｄの受け入れ態勢確認機能を利用しました．病院Ｄは医師リストから医師Ｆの余裕値確認機能を利用し，余裕値が 95 だったので救急センターに受け入れ可能の返事をしました．救急センターは救急車Ｃの病人確保機能を用いるために座標と搬送先（病院Ｄ）を伝えました．救急車Ｃは，確保・搬送をして，病院Ｄの受入機能を用いました．病院Ｄは医師Ｆの診察機能を使いました．

この課題には，必ずしも「正解」があるわけではありませんが，以下のことには気をつけてください．

- 登場人物などが全て書き出されているか
- 登場する機能や動詞がメソッドとして，名詞がフィールドとしてどこかに書かれているか
- オブジェクト同士の関連に過不足はないか（例えば，Ａさんは病院Ｄや医師Ｆを知らなくても大丈夫なはずです）
- できあがった図を使って，上のシナリオを説明できるか

以下はヒントです．必要なければ使わなくても構いません．