計算機プログラミング'95 ＃8

久野　靖*

1995.10.26

1　C++言語入門 (2)

1.1　「オブジェクト指向」について

さて、またまた守田さんからの感想で「オブジェクト指向というのが分からない」そうなので、その説明からしましょう。前回までのストーリーでは「パッケージ→抽象データ→オブジェクト」ときたのですが、それだとあんまり直接的でなかったかも知れません。

オブジェクト指向というのは、非常にひらたく言えば「現実世界で見られる『もの』に対応する概念をプログラム上でも作成し、それらの『もの』の集まりとしてソフトウェアを構成していく」ということです…これでもわかんないでしょうね。まあ授業で少しまとめて演説します。

1.2　動的配分 (dynamic dispatch)

さて、オブジェクト指向の利点の一つとしては、前回述べたように ADT が実現されるというのがあるのですが、今日は残りの利点つまり

- 動的配分による汎用コード
- 継承によるコードの共有

を取り上げます。(継承は次回にしようと思っていたのですが、皆様の様子から見るとじっくりやっても苦しいだけみたいですから、今日さっさと済ませて次回はおまけのテーマにします。)

まず、動的配分とは「オブジェクトの種類に関わらず似たようなオブジェクトには似たようなメッセージが送れる」「その結果の動作はオブジェクトの種類に応じたものになる」ということです。(また要演説ですね。)

問題は、C++のように強い型の言語では「オブジェクトの種類が違うと型も違うため、それらを一緒の変数には入れられない」という制約があることです。そこで、次の方法を取ります。

- クラスに親子関係を導入して、「類似したもの」を共通の親の子供 (ないし子孫) としてまとめる。
- 親クラスのポインタ型には、子孫のクラスのポインタ値が入れられる。

具体的には、前回「Circle」をオブジェクト (クラス) にしていましたが、これと独立に「Rect」を作る変わりに「DrawObj」(描けるもの) という親クラスを作り、Circle や Rect をその子孫にします。

*筑波大学大学院経営システム科学専攻

```
class DrawObj {
public:
  virtual void draw() { }
};
```

この DrawObj というクラスは実体変数は全くなく、draw というメソッドがあるだけです。しかもこのメソッドは何もしません! ({ }のところ。)

ではこれは何のためにあるのでしょう。draw というメソッドは、子供のクラスでそれぞれ固有のメソッドに置き換えます。そして、

```
  DrawObj *o = ...
  ...
  o->draw();
```

というのがあると、**現在 o に入っている『もの』に応じてその draw が呼ばれる**わけです (動的配分)。では子供のクラス。

```
class Circle : public DrawObj { ←ここで親を指定
private:
  Disp *dsp1;
  Window win1;
  int x1, y1, w1, h1;
public:
  Circle(Disp *d, Window m, int x, int y, int r);
  void draw();
};

Circle::Circle(Disp *d, Window m, int x, int y, int r) {
  dsp1 = d; win1 = m; x1 = x-r; y1 = y-r; w1 = h1 = r*2;
}

void Circle::draw() {
  XFillArc(dsp1->dsp(), win1, dsp1->gc(), x1, y1, w1, h1, 0, 360*64);
}
```

Rect も同様。

```
class Rect : public DrawObj {
private:
  Disp *dsp1;
  Window win1;
  int x1, y1, w1, h1;
public:
  Rect(Disp *d, Window m, int x, int y, int w, int h);
  void draw();
};

Rect::Rect(Disp *d, Window m, int x, int y, int w, int h) {
  dsp1 = d; win1 = m; x1 = x; y1 = y; w1 = w; h1 = h;
```

```
}

void Rect::draw() {
  XFillRectangle(dsp1->dsp(), win1, dsp1->gc(), x1, y1, w1, h1);
}
```

では本体を示します。

```
/* t81.c --- dynamic dispatch */

#include <X11/Xlib.h>
#include <X11/Xutil.h>

class Counter {
private:
  int value1;
public:
  Counter(int n);
  int add(int n);
  int value();
};

Counter::Counter(int n) { value1 = n; }
int Counter::add(int n) { return value1 += n; }
int Counter::value() { return value1; }

class Disp {
private:
  Display *dsp1;
  Screen *scr1;
  Window root1;
  unsigned long fg1, bg1;
  GC gc1, rgc1;
public:
  Disp::Disp();
  Display *dsp();
  Screen *scr();
  Window root();
  unsigned long fg();
  unsigned long bg();
  GC gc();
  GC rgc();
};

Disp::Disp() {
  dsp1 = XOpenDisplay(NULL);
  scr1 = DefaultScreenOfDisplay(dsp1);
  root1 = DefaultRootWindow(dsp1);
  fg1 = BlackPixelOfScreen(scr1);
  bg1 = WhitePixelOfScreen(scr1);
  gc1 = DefaultGC(dsp1, 0);
  XGCValues gcv;
  gcv.function = GXinvert; gcv.line_width = 3;
  rgc1 = XCreateGC(dsp1, root1, GCFunction|GCLineWidth, &gcv);
}

Display *Disp::dsp() { return dsp1; }
Screen *Disp::scr() { return scr1; }
Window Disp::root() { return root1; }
```

```
  unsigned long Disp::fg() { return fg1; }
  unsigned long Disp::bg() { return bg1; }
  GC Disp::gc() { return gc1; }
  GC Disp::rgc() { return rgc1; }

// DrawObj 類はここに入る。

  class DrawWin {
  private:
    Disp *dsp1;
    Window win1;
    Counter *cnt1;
    int exit1;
    DrawObj *objs1[100];
    int objuse1;
  public:
    DrawWin(Disp *d, int x, int y, int w, int h, Counter *c);
    void destroy();
    int handle(XEvent *ev);
    void drawall();
  };

  DrawWin::DrawWin(Disp *d, int x, int y, int w, int h, Counter *c) {
    exit1 = False;
    cnt1 = c;
    dsp1 = d;
    objuse1 = 0;
    win1 = XCreateSimpleWindow(d->dsp(), d->root(),x,y,w,h,2,d->fg(), d->bg());
    XSelectInput(d->dsp(), win1, ButtonPressMask|KeyPressMask|ExposureMask);
    XMapWindow(d->dsp(), win1);
  }

  void DrawWin::destroy() {
    cnt1->add(-1); XDestroyWindow(dsp1->dsp(), win1); exit1 = True;
  }

  int DrawWin::handle(XEvent *ev) {
    if(exit1 || ev->xany.window != win1) return False;
    if(ev->type == KeyPress) {
      char buf[20];
      if(XLookupString(&ev->xkey, buf, 20, 0, 0) == 1 && buf[0] == 'q') destroy(); }
    else if(ev->type == Expose)
      drawall();
    else if(ev->type == ButtonPress) {
      DrawObj *o;
      if(ev->xbutton.state&ShiftMask)
        o = new Rect(dsp1, win1, ev->xbutton.x-20, ev->xbutton.y-20, 40, 40);
      else
        o = new Circle(dsp1, win1, ev->xbutton.x, ev->xbutton.y, 20);
      o->draw(); objs1[objuse1] = o; ++objuse1; }
    return True;
  }

  void DrawWin::drawall() {
    XClearWindow(dsp1->dsp(), win1);
    for(int i = 0; i < objuse1; ++i) objs1[i]->draw();
  }

  main() {
```

```
  Counter *c = new Counter(3);
  Disp *d = new Disp();
  DrawWin *a[3];
  for(int i = 0; i < 3; ++i) a[i] = new DrawWin(d, 100, 100, 400, 200, c);
  while(c->value() > 0) {
    XEvent ev;
    XNextEvent(d->dsp(), &ev);
    for(i = 0; i < 3; ++i)
      if(a[i]->handle(&ev)) break;
  }
}
```

演習 1 このまま動かせ。

演習 2 三角形オブジェクトを作成し追加せよ。(塗らなくてよい。)

1.3 実現の継承

上のではインタフェースのみを継承していたが、dsp1 や win1 などの実体変数は共通していましたね? そこで、これらを親クラスにまとめることを考える。

```
class DrawObj {
protected:
  Disp *dsp1;
  Window win1;
  int x1, y1, w1, h1;
public:
  DrawObj(Disp *d, Window m);
  virtual void draw() { }
};

DrawObj::DrawObj(Disp *d, Window m) {
  dsp1 = d; win1 = m;
}
```

protected: というのは、外からはアクセスできないが、子孫のクラスからはアクセスできるという意味。子孫のクラスにおける実体変数群は親クラスの実体変数をすべて集めたものになる。そして、実体変数があると普通はコンストラクタも必要。で、子クラスのコンストラクタからは親クラスのコンストラクタを呼び出すのが普通。

```
class Circle : public DrawObj {
public:
  Circle(Disp *d, Window m, int x, int y, int r);
  void draw();
};

Circle::Circle(Disp *d, Window m, int x, int y, int r) : DrawObj(d, m) {
  x1 = x-r; y1 = y-r; w1 = h1 = r*2;
}
```

```
void Circle::draw() {
  XFillArc(dsp1->dsp(), win1, dsp1->gc(), x1, y1, w1, h1, 0, 360*64);
}

class Rect : public DrawObj {
public:
  Rect(Disp *d, Window m, int x, int y, int w, int h);
  void draw();
};

Rect::Rect(Disp *d, Window m, int x, int y, int w, int h) : DrawObj(d, m) {
  x1 = x; y1 = y; w1 = w; h1 = h;
}

void Rect::draw() {
  XFillRectangle(dsp1->dsp(), win1, dsp1->gc(), x1, y1, w1, h1);
}
```

練習 3 さっきのクラスをこの形に直せ。

2 親クラスにおけるメソッド

ところで、データ構造が共通ということは、それに関する操作にも共通部分が存在してよいということ。たとえば、「移動」や「リサイズ」という操作をできるようにするには、各オブジェクトの位置や大きさをアクセスしたり変更したりできるようにしたい。そういうメソッドは DrawObj につけておけばよい。

```
class DrawObj {
protected:
  Disp *dsp1;
  Window win1;
  int x1, y1, w1, h1;
public:
  DrawObj(Disp *d, Window m);
  virtual int getx();
  virtual int gety();
  virtual int getw();
  virtual int geth();
  virtual void move(int x, int y);
  virtual void resize(int w, int h);
  virtual int hit(int x, int y);
  virtual void draw() { }
  virtual void drawrubber();
};

DrawObj::DrawObj(Disp *d, Window m) {
  dsp1 = d; win1 = m;
```

```
}

int DrawObj::getx() { return x1; }
int DrawObj::gety() { return y1; }
int DrawObj::getw() { return w1; }
int DrawObj::geth() { return h1; }
void DrawObj::move(int x, int y) { x1 = x; y1 = y; }
void DrawObj::resize(int w, int h) { w1 = w; h1 = h; }

int DrawObj::hit(int x, int y) {
  return x1 <= x && x <= x1+w1 && y1 <= y && y <= y1+h1;
}

void DrawObj::drawrubber() {
  XDrawRectangle(dsp1->dsp(), win1, dsp1->rgc(), x1, y1, w1, h1);
}
```

これらを利用してオブジェクトの移動を行うように直した DrawWin を示す。

```
class DrawWin {
private:
  Disp *dsp1;
  Window win1;
  Counter *cnt1;
  int exit1;
  DrawObj *objs1[100], *moving1;
  int objuse1;
  int dx1, dy1;
public:
  DrawWin(Disp *d, int x, int y, int w, int h, Counter *c);
  void destroy();
  int handle(XEvent *ev);
  void drawall();
};

DrawWin::DrawWin(Disp *d, int x, int y, int w, int h, Counter *c) {
  exit1 = False;
  cnt1 = c;
  dsp1 = d;
  objuse1 = 0;
  moving1 = NULL;
  win1 = XCreateSimpleWindow(d->dsp(), d->root(),x,y,w,h,2,d->fg(), d->bg());
  XSelectInput(d->dsp(), win1, ButtonMotionMask|ButtonReleaseMask
    |ButtonPressMask|KeyPressMask|ExposureMask);
  XMapWindow(d->dsp(), win1);
}

void DrawWin::destroy() {
  cnt1->add(-1); XDestroyWindow(dsp1->dsp(), win1); exit1 = True;
}

int DrawWin::handle(XEvent *ev) {
  if(exit1 || ev->xany.window != win1) return False;
  if(ev->type == KeyPress) {
    char buf[20];
    if(XLookupString(&ev->xkey, buf, 20, 0, 0) == 1 && buf[0] == 'q') destroy(); }
```

```
  else if(ev->type == Expose)
    drawall();
  else if(ev->type == ButtonPress && ev->xbutton.button == Button1) {
    DrawObj *o;
    if(ev->xbutton.state&ShiftMask)
      o = new Rect(dsp1, win1, ev->xbutton.x-20, ev->xbutton.y-20, 40, 40);
    else
      o = new Circle(dsp1, win1, ev->xbutton.x, ev->xbutton.y, 20);
    o->draw(); objs1[objuse1] = o; ++objuse1; }
  else if(ev->type == ButtonPress && ev->xbutton.button == Button2) {
    for(int i = 0; i < objuse1; ++i)
      if(objs1[i]->hit(ev->xbutton.x, ev->xbutton.y)) {
        moving1 = objs1[i]; moving1->drawrubber();
        dx1 = moving1->getx() - ev->xbutton.x;
        dy1 = moving1->gety() - ev->xbutton.y;
        break; }
  }
  else if(ev->type == MotionNotify && ev->xbutton.button == Button2) {
    if(moving1) {
      moving1->drawrubber();
      moving1->move(ev->xbutton.x+dx1, ev->xbutton.y+dy1);
      moving1->drawrubber(); }
  }
  else if(ev->type == ButtonRelease) {
    if(moving1) {
      moving1->drawrubber(); drawall(); moving1 = NULL; }
  }
  return True;
}

void DrawWin::drawall() {
  XClearWindow(dsp1->dsp(), win1);
  for(int i = 0; i < objuse1; ++i) objs1[i]->draw();
}
```

ところで、移動中のラバーバンドが丸くないと気持ちが悪いですね？　それには`Circle`で`drawrubber`を用意すればよい。このように、子クラスで親クラスにあるメソッドを再定義することを「オーバライド」とよぶ。オブジェクト指向だとこのように子供クラスを追加してはそこにメソッドを増やして行くことで徐々にプログラムを成長させられる。

練習 4 これらの変更を混ぜて動かせ。

練習 5 `Circle`のラバーバンドを丸くしてみよ。

練習 6 自分の作ったもののラバーバンドも作れ。

練習 7 大きさ変更もできるようにしてみよ。

3　複合オブジェクト

ところで、丸と四角が接した新しいオブジェクトを作りたくなったとしよう。その場合、またぞろ`XDraw`なんとかを使わなくてもいい。なぜか？　それは、**既存のオブジェクトを組み合わせて使えばよいから。**

練習 8 丸四角オブジェクトを作成してみよ。

4 オブジェクトのたどり

今度は、たとえばプリントアウトを考えてみよう。つまり、図形の各要素を PostScript に変換してファイルに出力する (PostScript は覚えていますね?)。それには、DrawWin で「p」というコマンドを打った時に PS ファイルをオープンして用意し、各オブジェクトに「自分を書き出せ」といい、最後に後始末すればよい。完成版のプログラム一式を最後に示す。

```
/* t84.c --- print out */

#include <stdio.h>
#include <X11/Xlib.h>
#include <X11/Xutil.h>

class Counter {
private:
  int value1;
public:
  Counter(int n);
  int add(int n);
  int value();
};

Counter::Counter(int n) { value1 = n; }
int Counter::add(int n) { return value1 += n; }
int Counter::value() { return value1; }

class Disp {
private:
  Display *dsp1;
  Screen *scr1;
  Window root1;
  unsigned long fg1, bg1;
  GC gc1, rgc1;
public:
  Disp::Disp();
  Display *dsp();
  Screen *scr();
  Window root();
  unsigned long fg();
  unsigned long bg();
  GC gc();
  GC rgc();
};

Disp::Disp() {
  dsp1 = XOpenDisplay(NULL);
  scr1 = DefaultScreenOfDisplay(dsp1);
  root1 = DefaultRootWindow(dsp1);
  fg1 = BlackPixelOfScreen(scr1);
  bg1 = WhitePixelOfScreen(scr1);
  gc1 = DefaultGC(dsp1, 0);
  XGCValues gcv;
  gcv.function = GXinvert; gcv.line_width = 3;
  rgc1 = XCreateGC(dsp1, root1, GCFunction|GCLineWidth, &gcv);
}

Display *Disp::dsp() { return dsp1; }
Screen *Disp::scr() { return scr1; }
Window Disp::root() { return root1; }
unsigned long Disp::fg() { return fg1; }
```

```
unsigned long Disp::bg() { return bg1; }
GC Disp::gc() { return gc1; }
GC Disp::rgc() { return rgc1; }

class DrawObj {
protected:
  Disp *dsp1;
  Window win1;
  int x1, y1, w1, h1;
public:
  DrawObj(Disp *d, Window m);
  virtual int getx();
  virtual int gety();
  virtual int getw();
  virtual int geth();
  virtual void move(int x, int y);
  virtual void resize(int w, int h);
  virtual int hit(int x, int y);
  virtual void draw() { }
  virtual void drawrubber();
  virtual void postscript(FILE *f);
};

DrawObj::DrawObj(Disp *d, Window m) {
  dsp1 = d; win1 = m;
}

int DrawObj::getx() { return x1; }
int DrawObj::gety() { return y1; }
int DrawObj::getw() { return w1; }
int DrawObj::geth() { return h1; }
void DrawObj::move(int x, int y) { x1 = x; y1 = y; }
void DrawObj::resize(int w, int h) { w1 = w; h1 = h; }

int DrawObj::hit(int x, int y) {
  return x1 <= x && x <= x1+w1 && y1 <= y && y <= y1+h1;
}

void DrawObj::drawrubber() {
  XDrawRectangle(dsp1->dsp(), win1, dsp1->rgc(), x1, y1, w1, h1);
}

void DrawObj::postscript(FILE *f) {
  fprintf(f, "newpath %d %d moveto %d 0 rlineto\n", x1, y1, w1);
  fprintf(f, "0 %d rlineto -%d 0 rlineto closepath fill\n", h1, w1);
}

class Circle : public DrawObj {
public:
  Circle(Disp *d, Window m, int x, int y, int r);
  void draw();
  void postscript(FILE *f);
};

Circle::Circle(Disp *d, Window m, int x, int y, int r) : DrawObj(d, m) {
  x1 = x-r; y1 = y-r; w1 = h1 = r*2;
}

void Circle::draw() {
```

```
  XFillArc(dsp1->dsp(), win1, dsp1->gc(), x1, y1, w1, h1, 0, 360*64);
}

void Circle::postscript(FILE *f) {
  int cx = x1 + w1 / 2;
  int cy = y1 + h1 / 2;
  fprintf(f, "gsave %d %d translate %d %d scale\n", cx, cy, w1/2, h1/2);
  fprintf(f, "newpath 0 0 1 0 360 arc closepath fill grestore\n");
}

class Rect : public DrawObj {
public:
  Rect(Disp *d, Window m, int x, int y, int w, int h);
  void draw();
};


Rect::Rect(Disp *d, Window m, int x, int y, int w, int h) : DrawObj(d, m) {
  x1 = x; y1 = y; w1 = w; h1 = h;
}

void Rect::draw() {
  XFillRectangle(dsp1->dsp(), win1, dsp1->gc(), x1, y1, w1, h1);
}

class DrawWin {
private:
  Disp *dsp1;
  Window win1;
  Counter *cnt1;
  int exit1;
  DrawObj *objs1[100], *moving1, *res1;
  int objuse1;
  int dx1, dy1;
public:
  DrawWin(Disp *d, int x, int y, int w, int h, Counter *c);
  void destroy();
  int handle(XEvent *ev);
  void drawall();
};

DrawWin::DrawWin(Disp *d, int x, int y, int w, int h, Counter *c) {
  exit1 = False;
  cnt1 = c;
  dsp1 = d;
  objuse1 = 0;
  moving1 = NULL;
  res1 = NULL;
  win1 = XCreateSimpleWindow(d->dsp(), d->root(),x,y,w,h,2,d->fg(), d->bg());
  XSelectInput(d->dsp(), win1, ButtonMotionMask|ButtonReleaseMask
    |ButtonPressMask|KeyPressMask|ExposureMask);
  XMapWindow(d->dsp(), win1);
}

void DrawWin::destroy() {
  cnt1->add(-1); XDestroyWindow(dsp1->dsp(), win1); exit1 = True;
}

int DrawWin::handle(XEvent *ev) {
```

```
  if(exit1 || ev->xany.window != win1) return False;
  if(ev->type == KeyPress) {
    char buf[20];
    if(XLookupString(&ev->xkey, buf, 20, 0, 0) != 1)
      ;
    else if(buf[0] == 'p') {
      FILE *f = fopen("out.ps", "w");
      fprintf(f, "20 500 translate 1 -1 scale 0.5 setgray\n");
      for(int i = 0; i < objuse1; ++i) objs1[i]->postscript(f);
      fprintf(f, "showpage\n"); fclose(f); }
    else if(buf[0] == 'q')
      destroy();
  }
  else if(ev->type == Expose)
    drawall();
  else if(ev->type == ButtonPress && ev->xbutton.button == Button1) {
    DrawObj *o;
    if(ev->xbutton.state&ShiftMask)
      o = new Rect(dsp1, win1, ev->xbutton.x-20, ev->xbutton.y-20, 40, 40);
    else
      o = new Circle(dsp1, win1, ev->xbutton.x, ev->xbutton.y, 20);
    o->draw(); objs1[objuse1] = o; ++objuse1; }
  else if(ev->type == ButtonPress && ev->xbutton.button == Button2) {
    for(int i = 0; i < objuse1; ++i)
      if(objs1[i]->hit(ev->xbutton.x, ev->xbutton.y)) {
        moving1 = objs1[i]; moving1->drawrubber();
        dx1 = moving1->getx() - ev->xbutton.x;
        dy1 = moving1->gety() - ev->xbutton.y;
        break; }
  }
  else if(ev->type == MotionNotify && ev->xbutton.button == Button2) {
    if(moving1) {
      moving1->drawrubber();
      moving1->move(ev->xbutton.x+dx1, ev->xbutton.y+dy1);
      moving1->drawrubber(); }
  }
  else if(ev->type == ButtonPress && ev->xbutton.button == Button3) {
    for(int i = 0; i < objuse1; ++i)
      if(objs1[i]->hit(ev->xbutton.x, ev->xbutton.y)) {
        res1 = objs1[i]; res1->drawrubber();
        dx1 = res1->getx()+res1->getw() - ev->xbutton.x;
        dy1 = res1->gety()+res1->geth() - ev->xbutton.y;
        break; }
  }
  else if(ev->type == MotionNotify && ev->xbutton.button == Button3) {
    if(res1) {
      res1->drawrubber();
      int w1 = ev->xbutton.x + dx1 - res1->getx(); if(w1 < 10) w1 = 10;
      int h1 = ev->xbutton.y + dy1 - res1->gety(); if(h1 < 10) h1 = 10;
      res1->resize(w1, h1); res1->drawrubber(); }
  }
  else if(ev->type == ButtonRelease) {
    if(moving1) {
      moving1->drawrubber(); drawall(); moving1 = NULL; }
    if(res1) {
      res1->drawrubber(); drawall(); res1 = NULL; }
  }
  return True;
}
```

```
void DrawWin::drawall() {
  XClearWindow(dsp1->dsp(), win1);
  for(int i = 0; i < objuse1; ++i) objs1[i]->draw();
}

main() {
  Counter *c = new Counter(3);
  Disp *d = new Disp();
  DrawWin *a[3];
  for(int i = 0; i < 3; ++i) a[i] = new DrawWin(d, 100, 100, 400, 200, c);
  while(c->value() > 0) {
    XEvent ev;
    XNextEvent(d->dsp(), &ev);
    for(i = 0; i < 3; ++i)
      if(a[i]->handle(&ev)) break;
  }
}
```

練習 9 自分のオブジェクトもプリントアウトできるようにせよ。

課題 5 お絵描きプログラムを改良して、何らかの新しい機能を追加せよ。