第2章　Javaプログラムの作成

このプリントでは、Javaプログラムの開発にJDKとよばれるSun Microsystems社から無償で提供されるコマンドライン上の開発環境を用いる。JDKはいくつかのプログラムから成り立っているが、主に用いるのは、＿＿＿＿というJavaコンパイラと＿＿＿＿という中間コード実行プログラム（JVMエミュレータ）、それに＿＿＿＿＿＿＿＿というアプレット（後述）を実行するためのプログラムである。

統合開発環境（IDE）としては、Sunの Java Studio, Borland社のJBuilder, それにオープンソースのEclipse[1] などがある。IDEは、エディタ・コンパイラ・デバッガなどが統合された環境で、プログラムを迅速に開発することができる。画面上でボタンなどのGUI部品を配置することができるものもある。

2.1　コンパイルと実行

慣習により、新しいプログラミング言語を学習する時の最初に、画面に “Hello World!” と表示するだけのプログラムを作成する。

2.1.1　Javaアプリケーション

例題 2.1.1 まず、通常のアプリケーション（つまりアプレットでない） *Hello World* プログラムは次のような形になる。このファイルを好みのエディタ（メモ帳、秀丸、*Emacs* など）で作成する。

ファイル Hello0.java

```
public class Hello0 {
  public static void main(String args[]) {
    System.out.println("Hello World!");
  }
}
```

Hello0.java を *Java* 仮想機械（*JVM*）のコードにコンパイルするには、＿＿＿＿というコマンドを用いる。

```
> javac Hello0.java
```

コンパイルが成功すれば、同じディレクトリに *Hello0.class* というファイルができている。これが、*JVM* のコードが記録されているファイルである。このことを確認して、*Hello0* を ＿＿＿＿で実行する。

[1] http://www.eclipse.org/

```
> java Hello0
```

（*.class* はつけない）すると *Hello World!* と画面に表示される。

このプログラムの意味を簡単に説明する。**public class** Hello0 は *Hello0* という _____ を作ることを宣言している。（クラスなど、オブジェクト指向の概念の詳しい説明は、後回しにする。）ただし、*Java* ではクラス名（この場合 *Hello0*）[2] とファイル名（この場合 *Hello0.java*）の *.java* を除いた部分は _____。この場合はどちらも *Hello0* である。*main* の型も *C* 言語の *main* 関数の型（*int main(int argc, char** argv)*）とは異なるが、とりあえず、この形（**public static** void main(String args[])）のまま覚えておく。*public* や *static* というキーワードについては後述する。*System.out.println* は *C* 言語の *printf* に相当するメソッドで文字列を画面に出力する。

2.1.2 Java アプレット

例題 2.1.2 次に _____ と呼ばれる *WWW* のページの中で実行されるプログラムの書き方を紹介する。

まず、次のような *2* つのファイルを好みのエディタ（秀丸、メモ帳、*Emacs* など）で作成する。

ファイル *Hello.java*

```
import javax.swing.*;
import java.awt.*;

public class Hello extends JApplet {
  public void paint(Graphics g) {
    g.drawString("HELLO WORLD!", 50, 25);
  }
}
```

ファイル *Hello.html*

```
<html>
<head> <title> A simple program </title> </head>
<body>
<applet code="Hello.class" width="150" height="25"> </applet>
</body>
</html>
```

Hello.java が *Java* のプログラム（アプレット）である。*Hello.html* は、このプログラムを埋め込む方の *HTML* 文書である。とりあえず、今は *HTML* の意味を理解する必要はない。（*HTML* ファイルの名前は、クラス名や *Java* ファイル名と _____。）

Hello.java を *JVM* のコードにコンパイルするためには、やはり _____ というコマンドを用いる。

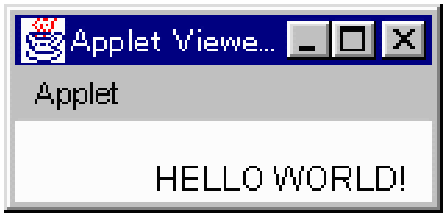
```
> javac Hello.java
```

コンパイルが成功すれば、同じディレクトリに *Hello.class* というファイルができている。これが、*JVM* のコードが記録されているファイルである。このことを確認して、*Hello.html* を _____ というプログラムで実行する。

```
> appletviewer Hello.html
```

[2] 正確には public なクラス名

すると次のような画面が表示されるはずである。



Netscape や *Internet Explorer* などの *WWW* ブラウザでもこの *Hello.html* を見ることができる。

2.1.3 Hello アプレット

これで、*Java* のアプレットを一つ作成できた。それでは、プログラムの意味を説明しよう。

最初の *2* 行の *import* 文は、大雑把にいえば *java.applet* と *java.awt* という二つの部品群（パッケージ, *package*）をプログラム中で使用することを宣言している。これ以外の形の *import* 文を用いることは当分ないので、これは呪文のようなものとしてこのまま覚えてしまった方が良い。（以降のプリント中の例では自明な場合、この *2* 行は省略する。）

次の *public class Hello extends JApplet* は、*JApplet* という _____ を _____ して、新しいクラス *Hello* を作ることを宣言している。*JApplet* クラスは、アプレットを作成する時の基本となるクラスで、アプレットとして振舞うための基本的なメソッドが定義されている。すべてのアプレットはこのクラスを継承して（つまり再利用して）定義する。このため、必要な部分だけを再定義すれば済む。

public はこのクラスの定義を外部に公開することを示している。逆に、公開しない場合は、*private* というキーワードを使う。はじめのうちは、*public* なクラスしか定義しないので、これも呪文のようなものとしてこのまま覚えてしまった方が良い。

Hello クラスは *JApplet* クラスの *paint* という名前のメソッドを上書きしている。クラスを継承する時は元のクラス（スーパークラスという）のメソッドを上書きすることもできるし、新しいメソッドやインスタンス変数を加えることもできる。

（参考）クラス名に使える文字の種類 Java では、クラス名に次の文字が使える（変数名、メソッド名なども同じ。）

> アンダースコア （“_”）, ドル記号（“$”）, アルファベット（“A”〜“Z”, “a”〜“z”）, 数字（“0”〜 “9”）, かな・漢字など（正確には Unicode 表 0xc0 以上の文字）

このうち数字は先頭に用いることはできない。

アルファベットの大文字と小文字は、_____。クラス名は大文字から始める、などのいくつかの決まりとまでは言えない習慣がある。public やvoid, for, if のように Java にとって特別な意味がある単語（_____）はクラス名などには使えない。

ドル記号とかな・漢字を用いることができるところが C や C++との違いである。

main はどこに行った? このプログラムには main 関数がない。アプレットは Web ブラウザの中で動作させることのできるプログラムの一部にすぎず、従来の意味でのプログラムではない。だからアプレットの main 関数にあたるものは Web ブラウザの main 関数であると言える。

2.1.4　JApplet クラス、Graphics クラス

アプレットにはいくつかのメソッドがあり、必要に応じてブラウザによって呼び出される。例えば、paint メソッドは、＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿に呼び出される。
他に主なものだけでも次のようなメソッドがある。

init メソッド　ブラウザが ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿に呼ばれる。
start メソッド　ブラウザが ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿に呼ばれる[3]。
stop メソッド　＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿に呼ばれる[4]。

```
public void paint(Graphics g)
```

という部分から、paint メソッドは ＿＿＿＿＿＿＿＿のオブジェクトを引数として受け取ること、戻り値はないことがわかる。public というキーワードがついていることと、class の定義の中に埋め込まれていることを除けば、C 言語の関数定義の方法と同じ書き方である。
Graphics クラスはいわば絵筆に対応するデータで、“絵の具の色” や “ 太さ” にあたるデータを要素（インスタンス変数）として持っている。このクラスのオブジェクトを使って画面上に文字や絵をかくことができる。Hello クラスでは、この Graphics クラスの drawString というメソッドを使って、“Hello World!” という文字を書いている。後ろの 50 と 25 は、表示する位置である。
このように Java ではオブジェクトのメソッドを呼び出すために、

オブジェクト ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

という形を用いる。また、インスタンス変数（メンバ）をアクセスする時は、

オブジェクト ＿＿＿＿＿＿＿＿

である。

問 2.1.3

1. *"Hello World!"*の部分を書き換えて、他の文字列を表示させよ。
2. *50, 25* の部分を書き換えて表示する位置を変更せよ。

2.1.5　アプレットと HTML

Hello.html の中でアプレットの実行に直接関係があるのは、

```
<applet code="Hello.class" width="150" height="50"> </applet>
```

の部分である。code=のあとに読み込みたいアプレットの名前（.class 付き）を書く、width と height はアプレットを表示するために確保する領域の幅と高さである。
なお、HTML の最近の規格では、applet タグは非推奨（deprecated）とされ、次のように object タグを用いることが推奨されている。

[3] init がよばれた後、あるいは他のページからアプレットのあるページに戻ってきた時など
[4] 他のページにジャンプする時など

```
<object codetype="application/java" classid="java:Hello.class"
        width="150" height="50">
</object>
```

しかし、当面はこの新形式をサポートしていないブラウザが多いので、このプリントでは旧来のapplet タグを用いて説明する。

次の例では _____というタグを使って、アプレットに HTML 側からパラメータ（引数）を渡している。このタグは <applet ... >と</applet>の間にはさむ。param タグに必要なものはパラメータの名前 （_____）とその値 （_____）である。プログラム中でパラメータの値を得るためには _____というメソッドを用いる。このメソッドは、パラメータの名前を引数として受け取って、そのパラメータの値を返す。

例題 2.1.4

ファイル Greeting.html

```
<html>
<head></head>
<body>
<applet code="Greeting.class" width="150" height="50">
  <param name="String" value="Bon Jour!">
</applet>
</body>
</html>
```

ファイル Greeting.java

```
import javax.swing.*;
import java.awt.*;

public class Greeting extends JApplet {
  public void paint(Graphics g) {
    String theArg;
    theArg = getParameter("String");
    g.drawString(theArg, 50, 25);
  }
}
```

上のアプレットを実行すれば、変数 *theArg* に、パラメータ *String* の値 *Bon Jour!* が入るので、画面に *"Bon Jour!"* と表示される。表示される文字列を *"Guten Tag!"* に変更したいときは、*Greeting.html* を書き換えれば済み、*Greeting.java* を再コンパイルする必要はない。

変数の宣言 上の例でわかるように変数の宣言は C と同様、

_____ 変数名 _

の形式で行なう。型名は int, String などのプリミティブ型か、クラス名である。ただし、C と違って、使用する前に宣言すれば必ずしも関数定義の最初に宣言する必要はない。変数への代入も C と同様 _____を使う。

インスタンス変数の宣言 ただし、上のプログラムでは画面を描き直すたびに getParameter が呼ばれ、効率は良くない。init メソッドで一度だけ getParameter が呼ばれるように書き直そう。

例題 2.1.5

ファイル Greeting.java（その 2）

```
import javax.swing.*;
import java.awt.*;

public class Greeting extends JApplet {
  String theArg;

  public void init() {
    // init はアプレットの初期化の時に一度だけ呼ばれる。
    theArg = getParameter("String");
  }

  public void paint(Graphics g) {
    g.drawString(theArg, 50, 25);
  }
}
```

このプログラムでは、theArg をインスタンス変数として宣言している。インスタンス変数の宣言は、クラス宣言の中に、メソッドの宣言と同じレベルに並べて書く。インスタンス変数はそのクラス中のすべてのメソッドから参照することができる。（ある意味で C 言語の大域変数と似ている。）

init メソッドの中でこの theArg に値を代入し、paint メソッドの中で参照している。このようにメソッドの中で自分自身のインスタンス変数を参照する時は、ピリオドを使った記法は必要ない。

インスタンス変数はオブジェクトが存在している間は値を保持している。これに対して、メソッドの中で宣言された変数（例えば、最初のバージョンの Greeting.java の theArg）の寿命はメソッド呼出しの間だけである。2 度め以降の呼び出しでも以前の値は保持していない。

Java のコメント　Java のコメントには C と同じ形式の /* と */ の間、という形の他にも、上の例のように // から行末まで、という形式も使える。（C++と同じ。）

問 2.1.6 *(JDKDIR)*[5]*/demo/Blink/*フォルダ、*(JDKDIR)/demo/NervousText/*の内容を適当なところにコピーして、*HTML* ファイルのパラメータを変更して実行せよ。

問 2.1.7 *(JDKDIR)/demo* フォルダなどから、その他の適当なできあいのアプレットを探してきて、パラメータを変えて実行せよ。

参考: エラーメッセージのリダイレクト　Java のソースファイルをコンパイルすると、エラーメッセージが大量に出力されて、最初の方のメッセージが見えないという事態が起こることがある。この場合は、エラーメッセージを別のファイル（ログファイル）に書き込んで（リダイレクトという）、あとでログファイルをメモ帳などで見るようにすると良い。

リダイレクトは次のような方法で行なう。

```
C:¥My Documents>javac ソースファイル 2> ログファイル
C:¥My Documents>javac -J-Djavac.pipe.output=true ソースファイル > ログファイル
```

上段が Windows NT/2000/XP の場合、下段が Windows 95/98/Me の場合である。

[5] *(JDKDIR)* は JDK をインストールしたディレクトリのことを指すことにする。これは JDK のバージョンにより異なる。

2.2 Java のグラフィクス （AWT）

Hello.java では、Graphics クラスの drawString メソッドを使って、画面に文字を表示したが、ここではこのクラスの他の描画メソッドを紹介する。

2.2.1 色とフォント

Graphics オブジェクトの色とフォントは次のメソッドで変更することができる。

`void setColor(Color c)` _____する。

`void setFont(Font f)` _____する。

例題 2.2.1

ファイル ColorTest.java

```
import javax.swing.*;
import java.awt.*;

public class ColorTest extends JApplet {
  public void paint(Graphics g) {
    g.setColor(Color.blue);
    g.drawString("Good Morning!", 20, 25);
    g.setColor(Color.orange);
    g.setFont(new Font("TimesRoman", Font.BOLD, 14));
    g.drawString("Good Afternoon!", 20, 50);
    g.setColor(Color.red);
    g.setFont(new Font("TimesRoman", Font.ITALIC, 14));
    g.drawString("Good Evening!", 20, 75);
  }
}
```

HTML ファイルの方ではアプレットの領域の高さを増やしておく（*height="100"*くらい）必要がある。実行すると次のようになる。

色を指定するためには、Color クラスのインスタンスを使う。上のプログラムのように、Color.blue, Color.red など定数として用意されているインスタンス[6]を用いる方法の他にも RGB 値を直接指定して新しい Color クラスのインスタンスを生成する方法もある。

[6] blue, red, orange の他に black, cyan, darkGray, gray, green, lightGray, magenta, pink, white, yellow が用意されている。

2.2.2　インスタンスの生成

一般に、あるクラスのインスタンスを生成するには、____という演算子を使う。new の次に _____ _____（constructor）という、クラスと同じ名前のメソッドを呼び出す式を書く。

Color クラスの場合、コンストラクタは 3 つのint 型の引数をとる。それぞれ 0 から 255 の範囲で赤（R）・緑（G）・青（B）の強さを表す。つまり、new Color(255,0,0)は純粋な赤を表すColor オブジェクトになる。g.setColor(Color.red);はg.setColor(_____);でも同じ意味になる。

Font クラスのコンストラクタは、フォントの種類を表す文字列（”TimesRoman”の他、”Courier”, ”Helvetica”, ”Dialog”, ”Symbol”のどれか）、スタイル（Font.BOLD（太字体）, Font.ITALIC（斜字体）, Font.PLAIN（通常の字体）の 3 つ）、サイズを表す整数、の 3 つの引数をとる。new Font("TimesRoman", Font.BOLD, 16) は、Times Roman という種類の太字体の 16 ドットのサイズのフォントである。

問 2.2.2 例題を改造して、いろいろな色・フォント・文字列を組み合わせを試せ。

2.2.3　図形の描画

Graphics クラスは、直線、長方形、多角形、楕円、円などさまざまな図形を描画するためのメソッドを持つ。

```
void drawLine(int x1, int y1, int x2, int y2)
```
(x1, y1) から (x2, y2) まで直線を引く。
```
void drawRect(int x, int, y, int w, int h)
```
左上の点が (x, y) で幅 w, 高さ h の長方形を描く。
```
void clearRect(int x, int, y, int w, int h)
```
左上の点が (x, y) で幅 w, 高さ h の長方形の領域をバックグラウンドの色で塗りつぶす。
```
void drawOval(int x, int y, int w, int h)
```
左上の点が (x, y) で幅 w, 高さ h の長方形に内接する楕円を描く。
```
void drawPolygon(int[] xs, int[] ys, int n)
```
(x[0], y[0])〜(x[n-1], y[n-1]) の各点を結んでできる多角形を描く。
```
void fillRect(int x, int, y, int w, int h)
```
左上の点が (x, y) で幅 w, 高さ h の長方形を描き塗りつぶす。

一般に、draw〜という名前のメソッドは内部を塗りつぶさず、fill〜は内部を塗りつぶす。

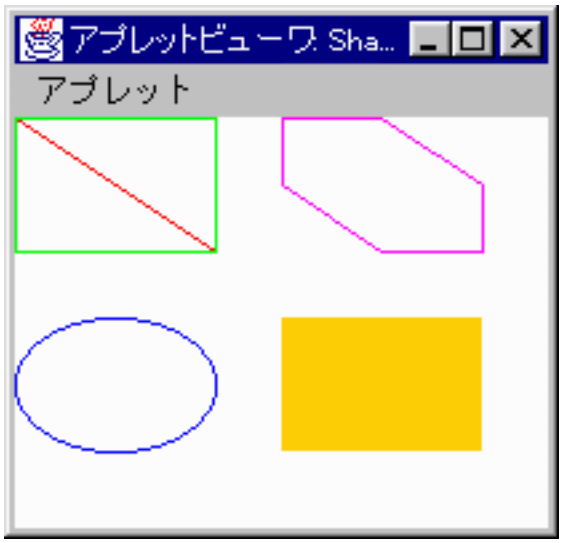
注意: Java のグラフィックスの座標系は左上の点が原点で、x 軸は通常と同じく右に向かって増えていくが、y 軸は数学で使われる座標軸と違って、下に向かって増えていく。単位はピクセル（画素）である。

例題 2.2.3

ファイル ShapeTest.java

```
import javax.swing.*;
import java.awt.*;

public class ShapeTest extends JApplet {
  public int[] xs = { 100, 137, 175, 175, 137, 100};
  public int[] ys = {   0,   0,  25,  50,  50,  25};
  public void paint(Graphics g) {
    g.setColor(Color.red);
    g.drawLine(0, 0, 75, 50);
    g.setColor(Color.green);
    g.drawRect(0, 0, 75, 50);
    g.setColor(Color.blue);
    g.drawOval(0, 75, 75, 50);
    g.setColor(Color.magenta);
    g.drawPolygon(xs, ys, 6);
    g.setColor(Color.orange);
    g.fillRect(100, 75, 75, 50);
  }
}
```



このクラスでは、2 つの変数 *xs, ys* を インスタンス変数として宣言している。これらのインスタンス変数は、メソッド *paint* の中で *drawPolygon* の引数として用いられている。

配列の宣言

```
public int[] xs = {100, 137, 175, 175, 137, 100};
```

は、C 言語では

```
int xs[] = {100, 137, 175, 175, 137, 100};
```

と書くべきところだが、Java ではどちらの書き方（[] の位置に注意）も可能である。[] は型表現の一部であるということを強調するため、Java では前者の書き方をする場合が多い。）

問 **2.2.4** *ShapeTest.java* の数値・色をなどをいろいろ変えて試せ。

問 **2.2.5** その他の *Graphics* クラスのメソッド:

```
void draw3DRect(int x, int y, int w, int h, boolean raised)
void drawArc(int x, int y, int w, int h, int angle1, int angle2)
void drawRoundRect(int x, int y, int w, int h, int rx, int ry)
void fillOval(int x, int y, int w, int h)
void fillPolygon(int[] xs, int[] ys, int n)
void fill3DRect(int x, int y, int w, int h, boolean raised)
void fillArc(int x, int y, int w, int h, int angle1, int angle2)
void fillRoundRect(int x, int y, int w, int h, int rx, int ry)
```

はどのような図形を描くか、試せ。

キーワード　JDK, javac, java, appletviewer, JApplet クラス, paint メソッド、init メソッド、start メソッド、stop メソッド、Graphics クラス、drawString メソッド、APPLET タグ、PARAM タグ、インスタンス変数、new, コンストラクタ、配列