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 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、javac という Java コンパイラーと java という中間コード実行プログラム（JVM エミュレーター）である。

統合開発環境（IDE）としては、Oracle 社（かつては Sun Microsystems 社） の NetBeans, IBM[1] によって開発された Eclipse[2], JetBrains 社の IntelliJ IDEA などがある。IDE は、エディター・コンパイラー・デバッガー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。

この章では JDK によって Java のプログラムを作成する方法を説明する。

2.1　コンパイルと実行

新しいプログラミング言語を学習するときの慣習により、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 コンソールアプリケーション

例題 2.1.1 まず、通常のアプリケーション（つまり後述の GUI を用いない）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ー（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.printf("Hello␣World!%n");
  }
}
```

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

```
> javac Hello0.java
```

[1] 現在は IBM とは独立した組織 Eclipse Foundation で開発されている。
[2] http://www.eclipse.org/

コンパイルが成功すれば、同じディレクトリーに `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 `java` で実行する。

```
> java Hello0
```

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

`javac` — Java のソースから中間コード（JVM コード）へのコンパイラー
`java` — JVM のエミュレーター

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。

```
> javac ソースファイル 2> ログファイル
```

Q 2.1.2 `Foo.java` という（無名パッケージの） Java のソースファイルを中間言語にコンパイルするためのコマンドを書け。
答: _____

Q 2.1.3 `Bar.class` という `main` メソッドを含む（無名パッケージの）クラスの Java の中間言語ファイルを実行するためのコマンドを書け。
答: _____

`Hello0.java` の意味を簡単に説明する。
`public class` `Hello0` は `Hello0` という _____（空欄 2.1.1）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、後述する。ただし、Java では、どんな簡単なプログラムでもクラスにしなければならないことになっているので、とりあえずこの形のまま使えば良い。）Java では public なクラス名（この場合 `Hello0`）とファイル名（この場合 `Hello0.java`）の `.java` を除いた部分は _____（空欄 2.1.2）。[3] この例の場合はどちらも `Hello0` でなければならない。この後の開ブレース（{）と対応する閉ブレース（}）の間がクラスの定義である。ここに変数（フィールド）や関数（メソッド）の宣言や定義を書く。

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。）このうち数字は先頭に用いることはできない。

[3] public でないクラス名に対しては、この規則は強制されない。

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（Unicode 表 0xc0 以上の文字）

Java は C 言語と同じようにアルファベットの大文字と小文字は、＿＿＿＿＿＿(空欄 2.1.3)。その他に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（キーワード）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。

Q 2.1.4 Java のクラス名として（文法的に）使える名前には○、使えないものには×をつけよ。

☐ 123Daaah　☐ Kagawa-U　☐ Drag'n'Drop　☐ 2_1　☐ Foo777
☐ Bar__　☐ HelloWorld!　☐ AreYouHappy?

Java アプリケーションの場合も C 言語と同じように、main という名前のメソッド（関数）から実行が開始されるという約束になっている。main メソッドの型は C 言語の main 関数の型（int main(int argc, char** argv)）とは異なる void main(String args[]) という型になっている。public や static というキーワード（修飾子）については後述する。とりあえず、この形（**public static void main(String args[])**）の形のまま使えば良い。

なお、String は Java の文字列の型である。C 言語と違い、String は char の配列ではない。しかし、文字列リテラル（String 型の定数）は、C 言語と同様二重引用符（" 〜"）に囲んで表す。

System.out.printf は C 言語の printf に相当するメソッドで文字列を変換指定に従って画面に出力する。つまりこのプログラムは、単に “Hello World!” という文字列を出力するプログラムである。%d（10 進整数）、%c（文字）、%x（16 進整数）、%s （文字列）、%f（浮動小数点数）、などの変換指定は C 言語の printf と同じように使用することができる。一方、%n は Java の変換指定に特有の書き方でシステムに依存する改行コード（Unix では ¥x0A, Windows では、¥x0D¥x0A）を表す。

Q 2.1.5 Java の main メソッドの型（つまりメソッド定義の「{」の手前の部分）を修飾子を含めて書け。
答: ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

2.1.2 Java GUI アプリケーション

例題 2.1.6 次に GUI (Graphical User Interface, グラフィカル・ユーザー・インターフェイス) を持つ Java プログラムの書き方を紹介する。このテキストでは、Swing という Java の標準的な GUI ライブラリーを紹介する。なお、JavaFX という Swing

より後に登場した Java の GUI ライブラリーも存在するが、Swing ほど普及していないこと、将来が不透明なこと、このテキストで扱う範囲の基本的な考え方は Swing と大差ないこと、などから、ここでは Swing を紹介する。

次のようなファイル（Hello.java）をエディター（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 JPanel {
  public Hello() {
    setPreferredSize(new Dimension(250, 50));
  }

  @Override
  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    g.drawString("HELLO␣WORLD!", 50, 25);
  }

  public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
      JFrame frame = new JFrame("こんにちは"); // タイトル部分
      frame.add(new Hello());
      frame.pack();
      frame.setVisible(true);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    });
  }
}
```

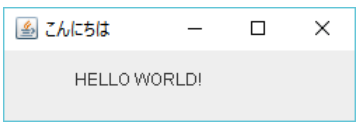
Hello.java を JVM のコードにコンパイルするためには、やはり javac コマンドを用いる。

```
> javac Hello.java
```

コンパイルが成功すれば、同じディレクトリーに 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 を java で実行する。

```
> java Hello
```

すると、右のような画面が表示されるはずである。



Javaのコメント　Javaのコメントには C と同じ形式の＿＿＿＿（空欄 2.1.4）と＿＿＿＿（空欄 2.1.5）の間、という形の他にも、＿＿＿＿（空欄 2.1.6）から行末まで、という形式も使える。（C++と同じ。最近の C の仕様（C99）でも //〜形式のコメントが使えるようになっている。）

2.2 Hello クラス

これで、Java の GUI アプリケーションを一つ作成し実行することができた。続いて、プログラム（Hello.java）の意味を説明する。

最初の 2 行の import 文は、javax.swing 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。*は、このパッケージのすべてのクラスを使用する可能性があることを示している。典型的な GUI アプリケーションの場合、この 2 行の import 文を用いることが多い。（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。）

> **詳細:** パッケージは OS のディレクトリーやフォルダーがファイルを階層的に整理するのと 同じように、クラスを階層的に管理する仕組みである。JPanel クラスの正式名称は、パッケージの名前を含めた javax.swing.JPanel なのであるが、これを単に JPanel という名前で参照できるようにするのに
>
> ```
> import javax.swing.JPanel;
> ```
>
> という import 文を使う。javax.swing というパッケージに属するクラスすべてをパッケージ名なしで参照できるようにするのが、
>
> ```
> import javax.swing.*;
> ```
>
> という import 文である。
>
> 自作のクラスを他のクラスから利用する場合は適切なパッケージに配置するべきである。自作のクラスをパッケージの中に入れるために package 文（後述）というものを使う。このテキストで紹介するような main メソッドを持つクラスや、サーブレットの場合は、他のクラスから利用するわけではないので、パッケージなしでも良いだろう。正確に言うと package 文がないときは、そのファイルで定義されるクラスは無名パッケージというパッケージに属することになる。

注意: Java のソース（.java）ファイルやクラス (.class) ファイルは、そのパッケージに対応するディレクトリーに配置することになっている。このテキストでは当面の間、無名パッケージのクラスのみを作成するので、カレントディレクトリーの直下にソース（.java）ファイルもクラス（.class）ファイルも置かれていると想定する。他の場所に置くと、コンパイルや実行ができないことがある。

Java の既成のクラスを利用するためには、そのクラスが属するパッケージを調べて、それに応じた import 文を挿入する必要がある。もしくは、クラスをパッケージ名を含めたフルネームで参照する。ただし、java.lang という基本的なクラスを集めたパッケージは import しなくてもクラス名だけで使用できる。例えば String クラスは java.lang パッケージに属する。

次の public class Hello extends JPanel は、JPanel というクラスを継承して（つまり、ほんの少し書き換えて）、新しいクラス Hello を作ることを宣言している。（このとき、Hello クラスは JPanel クラスのサブクラス、逆に JPanel クラスは Hello クラスのスーパークラスと言う。）

JPanel クラスは、GUI アプリケーションを作成するときによく使われるクラスで、基本的なメソッドが既に定義されている。このため、必要な部分だけを追加定義すれば済む。

行の最初の public はアクセス修飾子と呼ばれ、このクラスの定義を外部に公開することを示している。（このテキストでは、はじめのうちは、public なクラスしか定義しない。）

次の、public Hello() { 〜 } はコンストラクターの定義である。コンストラクターはクラスと同じ名前の特別なメソッドである。他のメソッドの場合と異なり、コンストラクターの定義のときは戻り値の型は指定しない。public はやはりアクセス修飾子である。コンストラクターはインスタンスが生成されるときに初期化の役割を担う。このクラスの場合、コンストラクターはサイズの指定を行っているだけである。

さらに、Hello クラスは JPanel クラスの paintComponent という名前のメソッドを上書き（＿＿＿＿＿＿＿＿＿＿（空欄 2.2.1））している。クラスを継承するときは元のクラス（スーパークラス）のメソッドを上書きすることもできるし、新しいメソッドやフィールドを加えることもできる。このメソッドの中身については次節で説明する。paintComponent メソッドの定義の前の行の@Override は JDK5.0 から導入された＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿（空欄 2.2.2）というもので、スーパークラスのメソッドをオーバーライドすることを明示的に示すものである。これにより、スペリングミスなどによるつまらない（しかし発見しにくい）バグを減らすことができる。

最後に main メソッドの中では、“枠” である、JFrame 型のオブジェクトを生成し、この枠に Hello クラスの新しいインスタンス new Hello() を追加 (add) して表示している。

SwingUtilities.invokeLater(() -> { 〜 }); については、スレッドのところで説明する。

このテキストの以降のプログラム例では、コンストラクター名（Hello）の部分だけが変わるので、このような main メソッドの部分は掲載を割愛する。

Q 2.2.1 Qux という名前の JPanel を継承するクラスを定義するときの、import 文と public class に続く 3 ワードを書け。
答: public class ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

2.3 Graphics クラスのメソッド

JPanel にはいくつかのメソッドがあり、必要に応じて呼び出される。例えば、paintComponent メソッドは、画面を ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ (空欄 2.3.1) ときに呼び出される。

paintComponent メソッドは

```
public void paintComponent(Graphics g)
```

という部分から、Graphics 型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。

paintComponent メソッドは、その中で super.paintComponent(g) を呼び出している。super.〜はスーパークラスで定義されているメソッドを呼び出すための書き方である。super.paintComponent(g) は背景を再描画する働きがある。

Graphics クラスはいわば絵筆に対応するデータ型で、“絵の具の色” や “字の形” にあたるデータを構成要素（フィールド）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!” という文字列を書いている。後ろの 50 と 25 は、表示する位置である。

```
void drawString(String str, int x, int y)
```
座標 (x,y) に文字列 str を描画する。

2.4 メソッド呼出し

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. メソッド名 (引数 $_1$, . . . , 引数 $_n$)

という形を用いる。また、フィールド（インスタンス変数）をアクセスするときは、

オブジェクト. フィールド名

という書き方を用いる。前述したようにオブジェクトはいくつかのデータをまとめて一つの部品として扱えるようにした物であり、.（ドット）演算子は、オブジェクトの中から ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ (空欄 2.4.1) 演算子である。つまり、g.drawString(...) は、g という Graphics クラスのオブジェクトから drawString というメソッドを取り出して引数を渡す式である。Java のメソッドは必ずクラスの中で定義されている。そのため、同じオブジェクトのメソッドを呼出すなど特別な場合をのぞき、Java のメソッド呼出しには、このドットを使った記法が必要である。メソッドのドキュメントにはこの部分は明示されないので注意が必要である。

参考: .（ドット）演算子の前に書く値も、メソッド名の後の括弧の間に ,（コンマ）区切りで書く値も、どちらもメソッドに渡されるデータという意味では違いはないが、上述のようにイメージが異なる。. 演算子の前にあるのは “主語” で、括弧の間にある通常の引数は “目的語” のようなイメージである。

メソッドはクラスの中に定義されているので、同じ名前のメソッドが複数のクラスで定義されていて、同じ名前のメソッドでもクラスが異なれば実装が異なることがある。. 演算子の前のオブジェクトが、どのメソッドの実装を呼び出すかを決定する。

この点は**動的束縛**を説明するときに、より詳しく説明する。

Q 2.4.1 g という名前の変数が Graphics 型のオブジェクトのとき、座標 (12, 34) に “Thank You!” という文字列を表示するメソッド呼出しの式を書け。
答: _____

問 2.4.2

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

2.5　Java のグラフィクス （AWT）— 色とフォント

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。

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)	_____（空欄 2.5.1）する。
void setFont(Font f)	_____（空欄 2.5.2）する。

例題 2.5.1

ファイル ColorTest.java

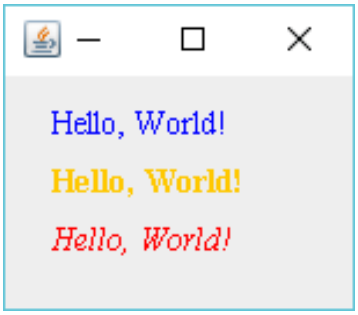
```
import javax.swing.*;
import java.awt.*;

public class ColorTest extends JPanel {
  public ColorTest() {
    setPreferredSize(new Dimension(150, 100));
  }

  @Override
  public void paintComponent(Graphics g) {
```

```
    super.paintComponent(g);
    String msg = "Hello,␣World!";
    g.setColor(Color.BLUE);
    g.setFont(new Font(Font.SERIF, Font.PLAIN, 14));
    g.drawString(msg, 20, 25);
    g.setColor(Color.ORANGE);
    g.setFont(new Font(Font.SERIF, Font.BOLD, 14));
    g.drawString(msg, 20, 50);
    g.setColor(Color.RED);
    g.setFont(new Font(Font.SERIF, Font.ITALIC, 14));
    g.drawString(msg, 20, 75);
  }

   ...  // main メソッドの定義は割愛
}
```



コンストラクターでは、領域の高さを増やしておく（100くらい）必要がある。実行すると左の図のようになる。

変数の宣言　変数の宣言はCと同様、

型名 (空欄 2.5.3) 変数名 (空欄 2.5.4)

の形式で行なう。型名は int, double などのプリミティブ型か、クラス名である。ただし、使用する前に宣言すればCと違って、必ずしも関数定義の最初に宣言する必要はない。初期値を指定するときは、変数名の後に= に続けて書く。

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（正確にはクラス変数）として用意されているインスタンス[4] 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。

> **Tips:** Javaのグラフィックス関数では標準ではアンチエイリアシングを行わないので、斜めの線の輪郭がギザギザに見えて美しくない。アンチエイリアシングをするには、描画の前に
>
> ```
> ((Graphics2D)g).setRenderingHint(
>    RenderingHints.KEY_ANTIALIASING,
>    RenderingHints.VALUE_ANTIALIAS_ON);
> ```
>
> という呼出しをしておけば良い。setRenderingHint は Graphics のサブクラスの Graphics2D クラスのメソッドである。JPanel の

[4] BLUE, RED, ORANGE の他に BLACK, CYAN, DARKGRAY, GRAY, GREEN, LIGHTGRAY, MAGENTA, PINK, WHITE, YELLOW が用意されている。

paintComponent メソッドに渡される引数は、実際には Graphics2D であるが、普段は Graphics クラスとして扱われるので、Graphics2D クラスのメソッドを利用するためにキャスト（型変換）を行っている。

2.6　クラスフィールドとクラスメソッド

＿＿＿＿＿＿＿＿＿＿[（空欄 2.6.1）]（クラス変数）はクラスに属するオブジェクトから共通にアクセスされる変数であり、＿＿＿＿＿＿＿＿＿＿[（空欄 2.6.2）]は、通常のフィールドにアクセスせずクラスフィールドだけにアクセスするメソッドである。どちらも、クラスによって決まるので、.（ドット）演算子の左にクラス名を書くことによってアクセスできる。以前に登場した System.out も System（正確に言うと java.lang.System）というクラスの out という名前のクラスフィールドである。

クラスメソッド・クラスフィールドのことを、それぞれスタティックメソッド・スタティックフィールドと呼ぶこともある。これは、クラスフィールドやクラスメソッドを定義するときに＿＿＿＿＿＿[（空欄 2.6.3）]という修飾子をつけるためである。API 仕様のドキュメントにも static と付記される。例えば、Color クラスのドキュメントの中では、

```
static Color BLACK
```

Math（java.lang.Math）クラスのドキュメントの中では、

```
static double cos(double a)
```

のように説明されている。これはそれぞれ使用するときには、Color.BLACK, Math.cos(0.1) のようにクラス名. メソッド名（またはフィールド名）の形に書かなければいけないということを示している。

なお、Java の static は、C 言語での static の使い方とはあまり関連がない。

また、Java アプリケーションで必ず定義する main メソッドも、スタティックメソッドでなければならない。

参考: Java 5.0 以降では static import という仕組みを利用することで、クラスフィールド・メソッドの前のクラス名を省略することができるようになった。例えば、プログラムの先頭に、

```
import static java.lang.Math.cos;  // cos 関数だけの場合、
// または
import static java.lang.Math.*;
// Math クラスのすべてのクラスフィールド・クラスメソッド
```

と書くと、単に cos(0.1) のように書くことができる。

Q 2.6.1 g という名前の変数が Graphics 型のオブジェクトのとき、以後の描画を緑色にするメソッド呼出しの式を書け。ただし、import java.awt.*; はしているが import static java.awt.Color.*; はしていないと仮定する。
答: ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

Q 2.6.2 円周率 π を表す定数は java.lang.Math クラスの中で

```
public static final double PI = 3.141592653589793;
```

と宣言されている。(final 修飾子は後述するが、この問に関しては無視してよい。)
（static import を使わないとき） 円周率を表す Java の式を書け。
答: _____

メソッド呼出しのまとめ
結局、Java のメソッド呼出しは、次の 3 通りの形式が存在することになる。

1. オブジェクト . メソッド名 (引数の並び)
   例: g.drawString("Hello", 20, 75)
2. クラス名 . メソッド名 (引数の並び)
   例: Math.cos(0)
3. メソッド名 (引数の並び) ··· ドットを使わない形
   例: setPreferredSize(new Dimension(150, 100))

このうち、1 が、もっとも普通のメソッド呼出しである。通常、API ドキュメントでは、次のようにメソッド名と引数の型だけが紹介される。

```
void drawString(String str, int x, int y)
```
座標 (x,y) に文字列 str を描画する。

しかし、使うときには、`g.drawString("HELLO WORLD!", 50, 25)` のように、オブジェクトを表す式とドットを前につける必要がある。
また、2 はクラスメソッドの呼出しである。API ドキュメントでは、クラスメソッドは次のように static というキーワードを伴って紹介される。

```
static double cos(double a)
```

この場合は、`Math.cos(Math.PI)` のように、クラス名（この場合は Math）とドットを前につける必要がある。
ここまではいずれにしても「.」（ドット）とその前に何かが必要である。ドットを使わず、3 のように書くのは、以下の 2 つの場合である。

- クラスメソッドに対し static import を使っている場合、
- 同じオブジェクトのメソッドを呼出す場合、

例えば、`import static java.lang.Math.*;` のような宣言があれば、クラスメソッドの Math.cos は単に `cos(PI)` のように呼出すことができる。
また、setPreferredSize メソッドは、JPanel クラス（正確には、JPanel のスーパークラスの JComponent クラス）に用意されているメソッドなので、JPanel を継承するクラスの（非 static な）メソッドからは、単に `setPreferredSize(new Dimension(100, 100))` のように呼出すことができる。

2.7　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、____(空欄 2.7.1) という演算子を使う。new の次に ____________________(空欄 2.7.2)（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。コンストラクターに必要な引数は各クラスにより異なるので API ドキュメントを調べる必要がある。また、ひとつのクラスが引数の型が異なる複数のコンストラクターを持つ場合もある。

Color クラスの場合、代表的なコンストラクターは 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(________________________(空欄 2.7.3));でも同じ色になる。

Font クラスのコンストラクターは、フォントの種類（Font.SERIF の他、Font.MONOSPACED, Font.SANS_SERIF, Font.DIALOG, Font.DIALOG_INPUT のどれかかプラットフォーム依存フォント名）、スタイル（Font.BOLD（太字体）, Font.ITALIC（斜字体）, Font.PLAIN（通常の字体）の 3 つの定数のどれか、または Font.BOLD | Font.ITALIC）、サイズを表す整数、の 3 つの引数をとる。new Font(Font.SERIF, Font.BOLD, 16) は、セリフ体の太字体の 16 ポイントのサイズのフォントである。

Q 2.7.1 g という名前の変数が Graphics 型のオブジェクトのとき、以後の文字列の描画に使うフォントの、サイズを 12 ポイント、種類を Font.MONOSPACED（等幅のフォント）、スタイルを通常、にするメソッド呼出しの式を書け。
答: ____

問 2.7.2 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。

2.8　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。

> **重要:** Java のクラスやメソッドは Java API 仕様というドキュメントにまとめられている。Java 8 の場合、`http://docs.oracle.com/javase/jp/8/docs/api/index.html` の左下のフレームからクラスを選択することによって、そのクラスに定義されているメソッド・フィールド・コンストラクターなどの仕様が上の `Graphics` クラスのメソッドの説明のような形式で示されている。今後は必要に応じてこのドキュメントを調べると良い。

例題 2.8.1

ファイル ShapeTest.java

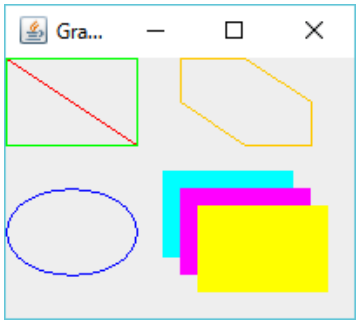
```
import javax.swing.*;
import java.awt.*;
import static java.awt.Color.*;

public class ShapeTest extends JPanel {
  public ShapeTest() {
    setPreferredSize(new Dimension(200, 150));
  }

  @Override
  public void paintComponent(Graphics g) {
    int[] xs = { 100, 137, 175, 175, 137, 100};
    int[] ys = {   0,   0,  25,  50,  50,  25};

    super.paintComponent(g);
    g.setColor(RED);
    g.drawLine(0, 0, 75, 50);
    g.setColor(GREEN);
    g.drawRect(0, 0, 75, 50);
    g.setColor(BLUE);
    g.drawOval(0, 75, 75, 50);
    g.setColor(ORANGE);
    g.drawPolygon(xs, ys, 6);
    g.setColor(CYAN);
    g.fillRect(90, 65, 75, 50);
```

```
    g.setColor(MAGENTA);
    g.fillRect(100, 75, 75, 50);
    g.setColor(YELLOW);
    g.fillRect(110, 85, 75, 50);
  }
}
```



drawLine, drawRect などはすべて Graphics クラスのメソッドであるので、Graphics クラスのオブジェクト g のメソッドを呼び出すため、＿＿＿＿（空欄 2.8.1）という形式で使用されていることに注意する。
このプログラムでは、2 つの変数 xs, ys を宣言している。これらの変数は、メソッド paint の中で drawPolygon の引数として用いられている。
このプログラムの実行結果は左の図のようになる。

配列の宣言　配列の宣言の

```
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 では前者の書き方が好まれる。

問 2.8.2　ShapeTest.java の数値・色などをいろいろ変えて試せ。

問 2.8.3　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、GUI アプリケーションを作成して試せ。

boolean 型 boolean 型は真偽値（＿＿＿＿(空欄 2.8.2) か ＿＿＿＿(空欄 2.8.3) の 2 つの値）を取り得る型である。

問 2.8.4 指定された文字が最初に出現する位置（最初から数えて何文字めか）を返すメソッド indexOf の使用例にならい、String（正式には java.lang.String）クラスのドキュメントで、次のような機能を持つメソッド:

1. 指定された文字が最後に出現する位置（最初から数えて何文字めか）を返す。
2. 文字列の m 文字目から n-1 文字目までの部分文字列を取り出す。

の使い方を調べて、実際にそれらを使用して、次の要件を満たす一つの Java コンソールアプリケーションを作成せよ。（いずれも最初の文字は 0 文字目と数える。空白も 1 文字と数える。）

1. 同じ文字列 msg の中で最後に 文字'e' が現れる位置を表示する。（33 文字目のはず）
2. 同じ文字列 msg の 11 文字目から 20 文字目を取り出して表示する。（"rown fox j"になるはず）

なお、文字リテラル（定数）は C 言語と同様、一重引用符に囲んで 'a' のように表す。
解答のテンプレート
ファイル StringExample.java

```
public class StringExample {
   public static void main(String[] args) {
      String msg = "The␣quick␣brown␣fox␣jumps␣over␣the␣lazy␣dog.";
      System.out.printf("文字␣'a'␣は␣%d␣文字目に現れます。%n",
                        msg.indexOf('a'));
       ...
   }
}
```

2.9 キーボードからの入力

Java でキーボードからの入力を扱うときは java.util.Scanner というクラスを用いる。次の例は、このクラスのオブジェクトの典型的な使用例である。
ファイル ScannerExample.java

```
import java.util.Scanner;

public class ScannerExample {
   public static void main(String[] args) {
      Scanner  sc = new Scanner(System.in);

      int i = sc.nextInt();
      System.out.printf("␣%d␣の␣2乗は␣%d␣です。%n", i, i * i);
   }
}
```

このクラスの nextInt というメソッドは、次の単語を int としてスキャンする。他に同じクラスのメソッドとして、 次の単語を double としてスキャンする nextDouble や、次の単語（単語の区切りは空白またはタブ文字）をスキャンする next、次の改行までスキャンする nextLine などのメソッドがある。

問 2.9.1 java.util.Scanner クラスの nextDouble, next, nextLine メソッドを使用したプログラムを作成せよ。

キーワード　JDK、class、javac、java、main メソッド、import、JFrame クラス、JPanel クラス、 継承、extends、オーバーライド、paintComponent メソッド、Graphics クラス、drawString メソッド、フィールド、クラスフィールド、クラスメソッド、new 演算子、 コンストラクター、Java API 仕様、配列、 boolean 型、