Javaへの招待

Javaへの招待

小島篤博*
ark@center.osakafu-u.ac.jp

1 はじめに

プログラミング言語Javaの名が知られるようになったのは、ここ1,2年のことです。このわずかな間に、現存するほとんどすべてのコンピュータ上で動作が可能となり、CやC++、BASICなど従来のプログラミング言語を圧倒するほどの支持を得るまでになりました。もちろん、昨今の爆発的なインターネットブームが追い風になっていることは確かですが、Java言語の背景や特徴を知っていただければ、この人気が単なる一過性のものではないことがおわかりいただけると思います。

本稿では、"Javaへの招待"ということで、このJava言語の背景や特徴を簡単に紹介します。さらに、SunSoft社が開発したJavaプログラミング環境である、Java WorkShop[1]を用いて、簡単なプログラム(タッチタイピング練習ソフト)を作成してみます。

2 Java登場の背景

Java言語が初めて姿を現したのは、1995年5月のことです[2]。一般にはプログラミング言語としてよりも、HotJavaという名の、全く新しい機能を備えたWWWブラウザとして紹介されました(図1)。まだ文章と絵を表示するだけのブラウザしか存在しなかった当時、HotJavaが見せつけてくれたのは、キャラクタが踊り、グラフが動き、三目並べができるダイナミックなページでした。しかもこれらは(HotJava自身も含めて!)、Javaという新しい汎用プログラミング言語で書かれていたのです。もしこれが単にブラウザをコントロールするだけの簡易言語であったなら、誰もこれほどまでに注目はしなかったでしょう[3]。

Javaが注目されたのは、汎用プログラミング言語として必要な機能をすべて備え、かつ生まれながらにしてネットワーク上を自由に飛び回ることが約束されていたためです。Javaで書かれたプログラムは、WWWサーバ上に置いて適当にリンクしさえすれば、世界中のどこからでもアクセスしてWWWブラウザ上にダウンロードし、実行することができるのです。このような形態のプログラムを、一般にアプレットといいます。

最初のうちはホームページを飾るアクセサリに過ぎませんでしたが、次第にJavaの特性を生かした様々なアプレットが作られるようになり、やがてNetscapeなど他のWWWブラウザもJavaに対応するようになりました。そもそもWWWブラウザそれ自体が新しい種類のソフトウェアであり、非常に短期間のうち

*大阪府立大学 総合情報センター助手

[1]総合情報センター 実習室2およびオープンスペースのUNIXワークステーションでも利用可能となっています。

[2]Sun社がJavaプロジェクトを開始したのは、1991年のことだそうです。私も当時、SunのBill Joy氏が、「C++の欠点を克服したプログラミング言語を開発中であり、コードネームはC++++−= (C-plus-plus-plus-plus-minus-equal)である :-)」とビデオで語っているのをみて、一体どんな言語だろうと興味を引かれたのを覚えています。

[3]その場合は、また世の中に特殊用途の言語が一つ増えるだけの話です。このような言語としては、Netscape社のJavaScriptがあり、比較してみると世間の待遇の違いがわかるでしょう。なお蛇足ですが、JavaScriptとJavaとは、言語仕様上まったくの別物ですので、混同しないように注意してください。

図 1: HotJava

に旧来のOSやアーキテクチャの境界を超えて広がっていったことは、Javaが普及する大きな要因となりました。

JavaがWWWと共に普及したのは、Javaの次のような特徴によるところが大きいと思われます。

ネットワーク透過性

Javaのプログラム(アプレット)を、WWWサーバからWWWブラウザに、ネットワークを越えて動的にダウンロードし、実行するしくみがある。これはWWWだけでなく、ソフトウェア流通の新しいインフラとなる可能性がある。

プラットフォーム独立性

Javaは、OSやプロセッサなどプラットフォームごとの差異を、バーチャルマシン(仮想機械)と呼ばれる層において吸収する。このため、単一のコードがすべてのプラットフォーム上で動作可能である。また、一旦ダウンロードされたコードを、特定のマシンに最適なコードへ2次的にコンパイルすることもでき[4]、パフォーマンス(実行性能)的にも配慮がなされている。

オブジェクト指向

ソフトウェアのコンポーネント化(部品化)の基盤となるオブジェクト指向技術をサポートしている。このため、Visual Basicなどで一般的になった、適切な機能・サイズのソフトウェア・コンポーネントをネットワーク上で転送したり、流通したりするのに有利である。

Javaとよく比較されるプログラミング言語としては、C++やVisual Basicなどが挙げられます。C++は、C言語の発展型として、ここ数年最もメジャーなプログラミング言語として利用されてきました。しかしながら、C++の言語仕様は複雑で不整合な部分が多く、その機能のすべてを習得するのは極めて困難

[4]このようなコンパイラを、**JIT (Just-In-Time) コンパイラ**といいます。

と言わざるをえません。一方Javaは、文法的にはC++によく似ていますが、同時にC++の複雑な部分を一切排除し、代わりにプログラマの負担を減らすような機能が盛り込まれています[5]。敢えてC++と似た文法を採用したのは、開発の経緯もありますが、すでにC++を経験しているプログラマたちのスムーズな移行を狙ってのことでしょう。

またVisual Basicは、Microsoft Windowsでの利用に限るなら、統一されたプログラミング/実行環境を提供してるといえます。しかしJavaが提供しようとしているのは、MacやUNIXを含むすべてのコンピュータ上でのプログラミング/ 実行環境の統一です。Visual Basicで書いたプログラムはWindowsでしか動きませんが、Javaで書きさえすれば、WindowsであれMacであれUNIXであれ、プラットフォームを問わずに動作することが可能になります。

これまでも、プラットフォームごとの違いを吸収した統一的な環境で、アプリケーションを動作させるというアイデア自体はありました。しかしながら、どうしても期待するような性能が出ないことと、普及が難しいことが予想されることなどから、まさ「絵に描いた餅」でしかありませんでした。ところが最近になって、マイクロプロセッサの性能向上とコンパイラ技術の進歩により、技術面での壁はほぼクリアされ、あとはJavaのようなユニバーサルなソフトウェア・アーキテクチャが、正しいタイミングで市場投入されるのを待つばかりという状態にあったわけです。技術だけをみると、Another Javaとなりそうな候補は他にもありましたが、この絶妙なタイミングと、新しいタイプのWWWブラウザを同時に発表する戦略が、Javaの運命を決定付けたといえます。

前置きが長くなりましたが、次章ではJava言語の特徴や機能について解説していきましょう。

3 Java言語の概要

3.1 Javaプログラムの種類

Javaのプログラムには、次のような種類があります。

スタンダローン・プログラム
: 従来のプログラムと同様、OSのシェルから起動するプログラム形態。GUI (Graphical User Interface)を全く持たない文字ベースのプログラム、GUIを持つウィンドウベースのプログラムのいずれも作成可能である。前述のHotJavaも、スタンダローンJavaプログラムの一例である。

アプレット Javaの特徴ともいえる、WWWブラウザ上にダウンロードされ、実行されるプログラム形態。
: アプレットを開発する場合は、必然的にGUIを意識したイベント・ドリブン型のプログラムを書くことになる。

このように、WWWブラウザ上のアプレットだけでなく、従来のプログラミング言語と同じように、シェルのコマンドラインから起動する形のプログラムを作ることができます。アプレットについては、後でプログラム例を用いて解説することにして、ここJava言語そのものの特徴について説明することにします。

3.2 Hello, World

まず簡単な例として、"Hello, World!" とだけ表示するプログラムを図2に示します。これを、`Hello.java`というファイル名でセーブしてください[6]。

---

[5]例をあげるなら、自動記憶領域管理(Garbage Collection)、単一のクラス階層、そしてポインタの排除など、C++がCとの互換性を引きずっていたために達成できなかったものばかりです。

[6]Javaでは、1クラス-1ファイルが基本です。ファイル名には、クラス名の後に'`.java`'を付けた名前を使います。

```
1: public class Hello {
2:   public static void main(String args[]) {
3:     System.out.println("Hello, World!");
4:   }
5: }
```

図 2: Helloプログラム

このようなスタンダローン・プログラムは、Javaコンパイラを使ってコンパイルし、Javaバーチャルマシン上で実行します。以下に、Sunが無料で配布しているJDK[7]を用いた実行例を示します。

```
$ javac Hello.java  ..... (1)
$ ls  ..... (2)
Hello.class    Hello.java
$ java Hello  ..... (3)
Hello, World!
$
```

まず、Hello.javaをJavaコンパイラ(コマンド名javac)でコンパイルします(1)。コンパイルが完了すると、Javaのバーチャルマシン・コード(ファイル名Hello.class)が生成されます(2)。そしてこれを、Javaバーチャルマシン(コマンド名java)で起動します(3)。"Hello, World!" と表示されたのがわかるでしょう。

では、このHelloプログラムを例にとりながら、Java言語の特徴や機能をみていきましょう。

3.3 クラスとオブジェクト

まず最初に注目していただきたいのは、全体を囲む class Hello { ... } というブロックです。これは、このプログラムが、Helloという1つの**クラス**からなっていることを表わしています。クラスというのは、簡単にいえばソフトウェアの部品の定義で、オブジェクト指向プログラミングに特有の概念です。Java言語では、クラスは最も重要なプログラムの構成単位で、整数や文字列などと同様、データ型として扱われます。このようなクラスを定義していくことが、すなわち、Javaプログラムを書くことと同義なのです。

クラスの定義は、新しいものを一から作るのではなく、既存のほかのクラスの機能を受け継ぐ形で行ないます。これを**継承**といいますが、一般にプログラムの開発効率や保守性を高めるために有効とされている機能です。

また、クラスと関連した概念に、**オブジェクト**というのがあります。整数や文字列などのデータ型に対して、99や "Hello, World!" などの実データが存在するのと同様に、クラスという型に対する実体をオブジェクトと呼んでいます。実行時には、1つのクラスに対して複数のオブジェクトが存在しているのが普通です。

3.4 データ型

ユーザによる定義が可能なクラスとは別に、言語仕様の中で最初から定義されているデータ型がいくつかあります。整数型(short, int, long)、文字型(char)、実数型(float, double)、論理型(boolean)、そしてバ

[7]Java Development Kit. SolarisやWindows95/NTで動作します。このほか多数のWSメーカが、自社のOSに移植して配布しているようです。

イト型(byte)がそうです。これらの型のデータ長は、異なるアーキテクチャ間での互換性を保つため、厳密に定められています[8]。また文字型は、国際化を考慮してunicodeを格納できる16bitとなっています。

3.5 メソッドとメンバ変数

一般的なプログラミング言語と同様、Javaにも**手続き**があります。ただし、すべての手続きは、いずれかのクラスに属することとなっており、名称も(手続きではなく)**メソッド**となっています。図2のプログラムでは、`main`というメソッドが定義されています。関数の返り値や引数の指定はC/C++と同様で、ここでは`String[]`型の引数をとり、値を返さない(`void`)メソッドあることが宣言されています。[9]

また、クラスの中には、メソッドと同様に変数を含めることもできます。このような変数をメンバ変数といいます。

3.6 制御文

`main`メソッドの本体は、`System.out.println("...");`という1文だけですが、Java言語には他にも様々な制御文が用意されており、そのほとんどはC/C++と同じです。

if文

`if` (*condition*) *then-clause* `else` *else-clause*

条件判定式*condition*が真であれば*then-clause*を、そうでなければ*else-clause*を実行します。

while文

`while` (*condition*) *body*

条件式*condition*が真である間、*body*を繰り返し実行します。while文の直前にラベル

label:

を置いておけば、何重にもネストしたwhile文の*body*から、

`break` *label*;

で一気に抜けることができます。

for文

`for` (*init-expr*; *condition*; *increment*) *body*

最初に*init-expr*を実行したのち、*condition*が真である間、*body*および*increment*を交互に繰り返し実行します。*init-expr*では、*body*の内部でのみ有効な変数を宣言することができます。

この他、多重分岐のためのswitch-case文、例外処理を扱うためのtry-catch文[10]などが用意されています。

[8]データ長とは、そのデータがメモリ上に占めるバイト数のことです。C/C++では、対象となるアーキテクチャにとって最高のパフォーマンスを出すために、データ長の定義は任意となっています。

[9]これが実際に端末に文字列を出力している部分であることは、一見してお分かりでしょう。実は、プログラムの起動には約束があって、指定されたクラスの`main`という名前のメソッドが呼び出されることになっているのです。

[10]try-catch文はC++にも導入されていますが、Javaでは例外処理を**義務化**するなど、ロバストなプログラム作りに向けてより一歩踏み出しています。

4 Java WorkShop

4.1 Java WorkShopの概要

Java WorkShopは、SunSoft社が開発したJavaプログラムの統合開発環境です。すでに世の中には、この他にもMicrosoft社のVisual J++や、Symantec社のVisual Cafeなど、いくつかの統合開発環境がリリースされていますが、いずれも従来のC/C++などの開発スタイルを踏襲したものとなっています。一方Java WorkShop (以下JWS)は、開発環境そのものがブラウザとなっていて、プロジェクト管理やGUIデザイン、ビルド、デバッグなどが、それぞれ専用のページを開いて行なうようになっています[11]。このため、ネットワーク上での分散開発などが比較的シームレスに行なえるようになっています。一方で、JWS自体もJavaで記述されているため、現状では性能的に少々苦しいものがあります。(ここは我慢しましょう。)

今回は、サンプルプログラムとして、タッチタイピング練習用のアプレットを作ります。名前は'JavaKey'とでもしておきましょう。JavaKeyの仕様は以下のとおりです。

- キーボードと同じキーの配列を画面に表示し、次にタイプすべきキーをハイライト表示でガイドする。
- 正しいキーが入力された場合、その文字をエコーバックして次の文字に進む。間違ったキーが入力されても、先に進めない。
- キー入力練習用の短文を、HTML(ホームページを記述する言語)から渡せるようにする。(こうしておけば、HTMLファイルを書換えるだけで、練習用の短文を変更できる。)

タッチタイピング練習用のソフトとしては、いささか機能不足ではありますが、ここはサンプルプログラムということでご勘弁ください。

では、早速JWSを起動してみましょう。Solarisにおけるコマンド名は`jws`です。Windows95の場合は、スタートメニューからJava WorkShopを選びます。(Solaris版JWSの起動画面を、図3に示します。)JWSには、よくまとまったチュートリアルが付属してきますので、まずこちらを一通り体験することをお勧めします。時間にして1時間程度あれば、JWSでの作業の流れを理解できるはずです。チュートリアルは、JWS起動時のページにある、'Tutorial'というキーワードをクリックすれば始めることができます[12]。

さて、JWSにおけるプログラム開発は、**プロジェクト**(project)、および**ポートフォリオ**(portfolio)という単位で管理します。プロジェクトは開発するプログラムごとに作成し、Javaソースファイルや GUIファイルなどのファイル群を管理します。ポートフォリオは、複数のプロジェクトを包含したより大きな単位で、プログラムの目的や用途に応じた分類をするためのものです。初期状態では、personal、jdk、awtの3つのポートフォリオが用意されています。このうちpersonalポートフォリオは、個人的なプログラムを作成する際に使います。

それでは早速、personalポートフォリオに新しいプロジェクトを作成してみましょう。まず、JWSのメニューバーから'Portfolio ▷ Choose'を選び、表示されるポートフォリオの一覧メニューから'personal'を選んでください。次に、同じくメニューバーから、'Project ▷ Create ▷ Applet...'を選んでください。アプレット作成のためのページが表示されるはずです。(図4)

ここで、それぞれの欄に必要事項を記入します。

[11] これは、コンピュータのデスクトップをすべてHTMLとJavaのコンビで統合するという近い将来の枠組みを先取りしたものといえるでしょう。

[12] ここで体験できるチュートリアルは英語版です。日本語版のチュートリアルもあることはあるのですが、JWS自体がまだ日本語未対応のため、Netscapeなどで見る必要があります。

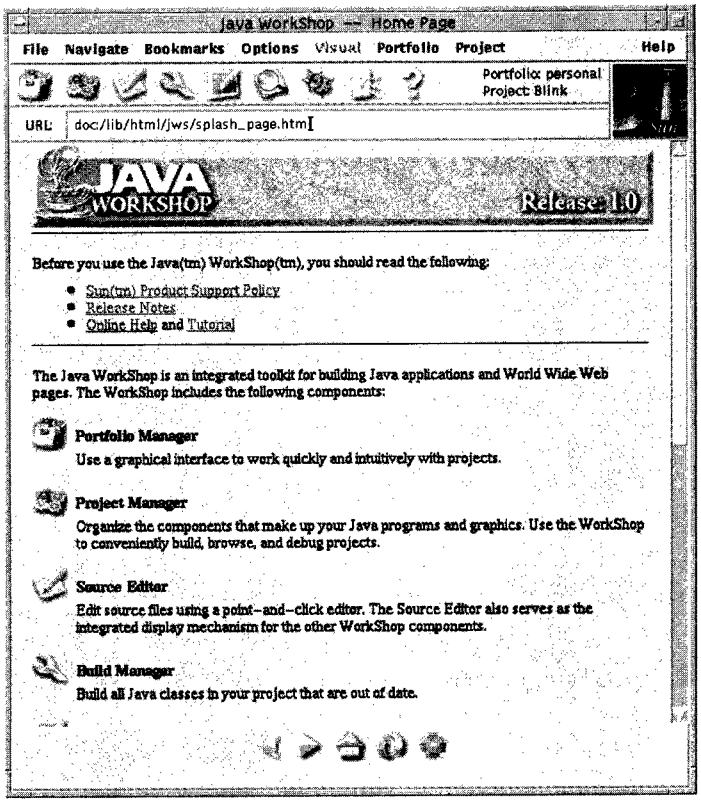


図 3: Java WorkShopの起動画面

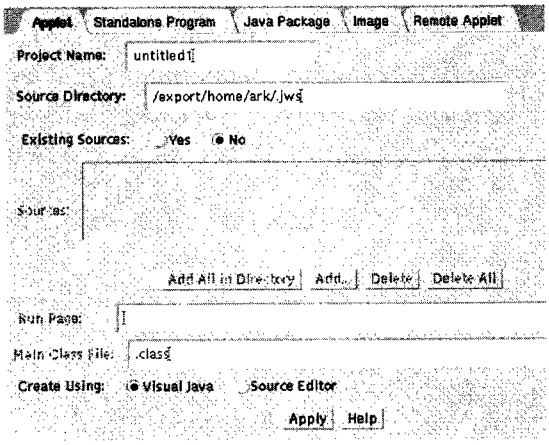


図 4: アプレット作成のページ

Project Name:	プロジェクト名。(ここでは'JavaKey'と入力してください。)
Source Directory:	プロジェクトを保存するディレクトリ名。
	(ここでは'*your-home*/.jws/JavaKey'と入力してください。)
Existing Sources:	既存のソースを利用するかどうか。
	今回は新規作成なので、Noを選択してください。
Create Using:	GUIを作るかどうか。'Visual Java'を選択してください。

入力が完了したら、Applyボタンを押してください。GUIをデザインするためのVisual Javaページが表示されるはずです。

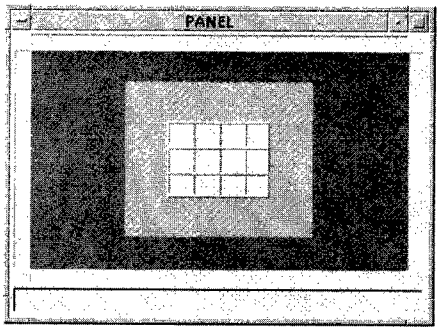
4.2 GUIのデザイン

Visual JavaはアプレットのGUIをデザインするためのツールで、アプレット画面のレイアウトを行なうウィンドウ(図5.a)と、画面上のコンポーネント(GUI部品)の各種属性を設定するページ(図5.b)から構成されています。一般にGUIのデザインは、最初にコンポーネントのレイアウトを決め、続いて各コンポーネントの属性を設定する、という流れになっています。このやり方はVisual Javaでも全く同様で、まずVisual Javaページにあるパレット(コンポーネントが2列に並んでいる領域)から、ボタンやラベルなどのコンポーネントを選択して、レイアウトウィンドウ上に配置します。そしてこのコンポーネントの属性を、Attribute Editorで設定します。

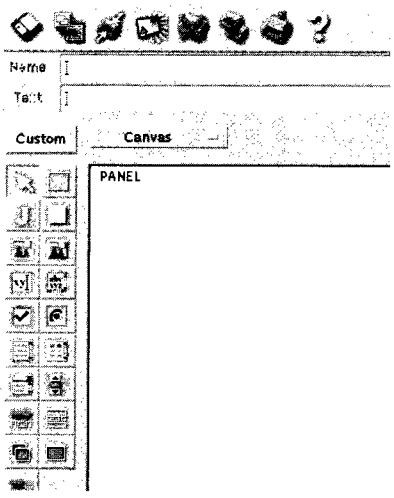
では早速、JavaKeyのキーボードのデザインをはじめましょう。以下の手順に従ってください。

1. パレット上のテキストラベル  をクリックしてください。アイコンが窪んだ形になり、選択されたことを表わします。

2. 次に、レイアウトウィンドウのグリッド(碁盤の目)の一番左上のマスをクリックします。すると、先ほど選択したテキストラベルが、そのマスの中に配置されるはずです。(図6) なお、このグリッドは、あくまで画面のレイアウトを決める補助をするためのものであり、アプレット実行時には表示されません。



a. アプレット レイアウトウィンドウ



b. Visual Java ページ

図 5: Visual Java

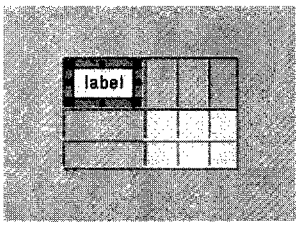


図 6: コンポーネントの配置

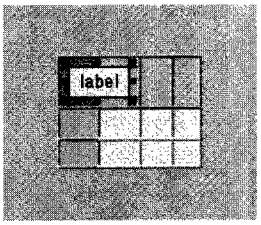


図 7: コンポーネントのサイズ変更

3. コンポーネントの大きさは、水平および垂直方向にいくつのスパンを占めるかで決まります。これは、コンポーネントの周囲8方向に配列されている、小さな黒い矩形をドラッグすることで変えることができます。(図7)

4. ここでVisual Javaページの方を見てみると、Nameフィールドと Textフィールドに、それぞれ "label1"、"label" という文字列が表示されているのがわかります。Nameというのは、このコンポーネントがJavaプログラムから参照されるときの名前です。またTextは、画面上に表示されるテキストラベルの文字列です。これらのフィールドは、必要に応じて書換えることができます。

   ここでは、Nameフィールドに "key1"、Textフィールドに "1" と入力してください。レイアウトウィンドウ上のテキストも、すぐに更新されるはずです。

5. 次に、コンポーネントの属性を変更します。各コンポーネントには、先ほどの名前(name)や文字列(text)のほか、文字色(foreground)や背景色(background)、フォント(font)などの属性があり、Javaプログラムの実行中に変更したり、Attribute Editorを用いて初期値を決めることができます。Attribute Editorを表示するためには、Visual Javaページのツールバーにある、Attribute Editorアイコンをクリックします。Attribute Editorは、図8のようになっています。

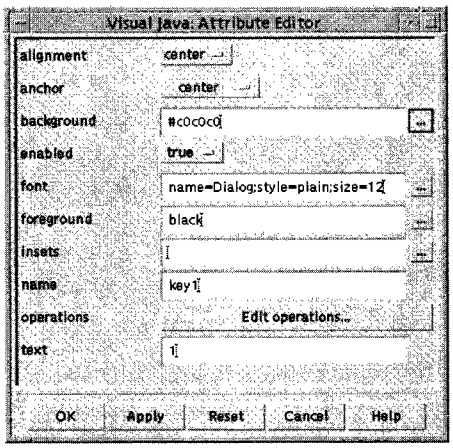


図 8: Attribute Editor

6. では、先ほど配置したラベルの背景色を白にしてみましょう。Attribute Editorの backgroundフィールドに、"white" と入力してみてください。レイアウトウィンドウ上のラベルの背景色が、白に変わるはずです。

7. このようにして、キーボードのレイアウトを作っていきます。最終的なレイアウト画面は、図9のようになります。注意点としては、キーの並びが列によって互い違いになるように、レイアウトを工夫

していることと、スペースキーの下に水平線(Labeled Bar)を入れたことです。グリッドの行や列を増やすには、Control +矢印キーを押します。

また、各コンポーネントの属性は、表1のように設定します。

表 1: コンポーネント属性表

name	text	background	name	text	background
key0...key9	0...9	white	keyA...keyZ	A...Z	white
keyMinus	-	white	keySColon	:	white
keyEqual	=	white	keyQuote	'	white
keyBSlash	\	white	keyComma	,	white
keyLP	[	white	keyPeriod	.	white
keyRP	]	white	keySlash	/	white
keyLShift	Shift	white	keySpace		white
keyRShift	Shift	white			
requestField	type me	#c0c0c0	label50	Enter =>	#c0c0c0
responseField		#c0c0c0			

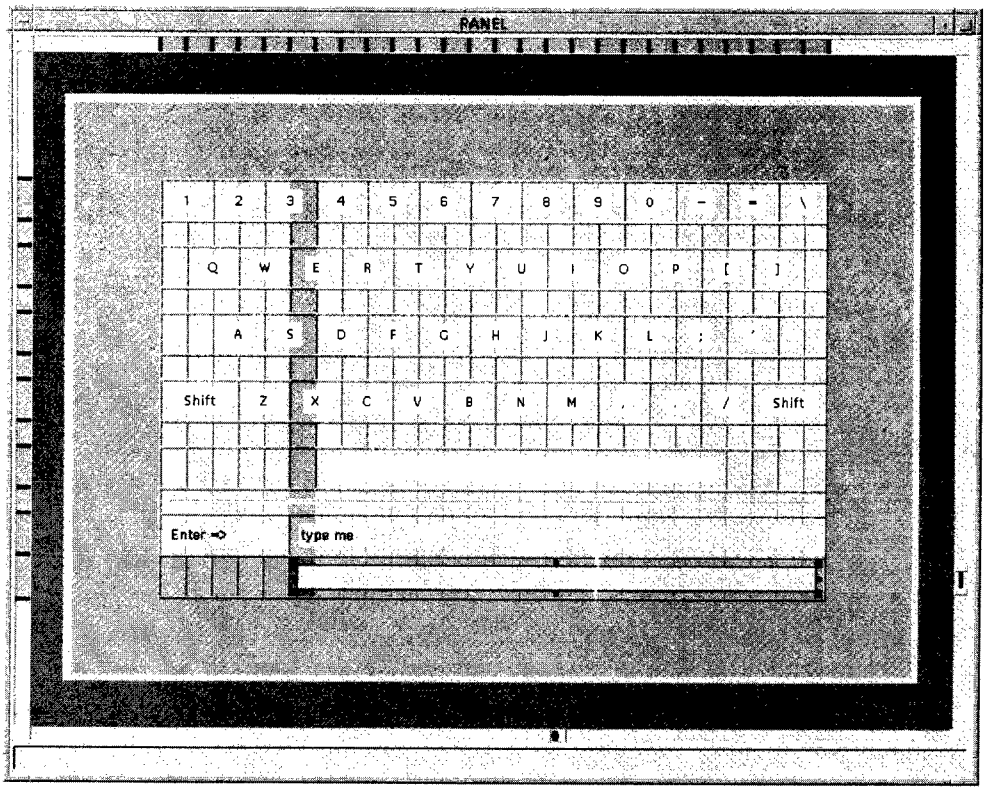


図 9: レイアウト完成図

8. これで、Visual JavaによるGUIデザインは終わりです。ツールバーにあるセーブ/ソース生成アイコン  をクリックしてください。これによって、デザインしたアプレットのGUIが保存され、Javaソースプログラムのスケルトン(骨組み)が生成されます。

次のセクションでは、ここで生成されたJavaソースプログラムに、JavaKey独自の処理を追加していきます。

4.3 コーディング

JWSは、Visual JavaによるGUIデザインの保存時に、GUIをコントロールするJavaのソースプログラムを生成することができます。といっても、自動生成されたプログラムは、Visual Javaで決めたレイアウトに従ってコンポーネントを表示するだけの機能しかありません。アプレット独自の処理を行なうためには、しかるべき場所にJavaコードを追加する必要があります。

JWSが生成するソースファイルは、次の3本です。

`JavaKey.java`
: JavaKeyアプレットのイベント処理部が含まれている。さまざまなイベントに対する処理を、ユーザが記述する。

`JavaKeyMain.java`
: JavaKeyアプレットの本体が定義されている。ユーザが直接編集してはいけない。

`JavaKeyRoot.java`
: 実行時にGUIコンポーネントを生成する。ユーザが直接編集してはいけない。

このうちユーザが編集する必要があるファイルは、`JavaKey.java`だけです。

実際の編集作業に入る前に、アプレットの処理について簡単に説明しておきます。一般にJavaアプレットは、キー入力やマウス操作などのイベントをきっかけにして、特定のプログラムコードを実行するようなしくみになっています[13]。この部分のスケルトンは、アプレット名と同じ名前をもつクラスに生成されます。このクラスのメソッド(一部)を表2に示します。

表 2: JavaKeyクラスのメソッド

メソッド	処理の概要
JavaKey	JavaKeyオブジェクトの生成時に呼び出される[14]。
initRoot	GUIコンポーネントを生成するために呼び出される。
startGroup	アプレットの実行開始時に呼び出される。
stopGroup	アプレットの実行停止時に呼び出される。
handleEvent	イベント処理を、適当なメソッドに振り分ける。
mouseDown	マウスが押されたときに呼び出される。
mouseDrag	マウスがドラッグされたときに呼び出される。
keyDown	キーが押されたときに呼び出される。

[13]このような形態を、**イベント・ドリブン(Event-Driven)型**のプログラムといいます。X Window SystemやWindowsなど、ウィンドウシステムを利用するGUIプログラムは、必然的にイベント・ドリブン型です。

[14]オブジェクトを初期化するための特別なメソッドを、**コンストラクタ**といいます。コンストラクタは、クラスと同じ名前を持ちます。

JavaKeyクラスの構成をみると、何らかのイベントに際して呼び出されるメソッドがあらかじめ用意されており、ユーザは基本的に適当なメソッドに処理を書き加えればよいことがわかります[15]。

では、JavaKeyで追加(修正)した部分について説明していきましょう。

- メンバ変数

KeyEntry keyTable[ ]

入力する文字と、キーボード上のキーラベルとの対応表です。たとえば、'A'の入力をガイドするためには、A キーと右 Shift キーの2つをハイライト表示にする必要があります。

なお、対応表のエントリは、KeyEntryクラスとして別に定義しています。

String stringToType

タッチタイピングの練習台となる文字列です。JavaKeyアプレット起動時に、HTMLからパラメータとして渡されるようにします。

int stringIndex

stringToTypeの何文字目まで進んだかを記憶するための変数です。

int lastKeyIndex

1文字前に入力されたキーの、keyTableに対するインデクスです。次のキーが入力されたとき、ハイライト表示を元に戻すために必要になります。

- メソッド

void startGroup()

keyTableを初期化し、HTMLから渡されたパラメータをstringToTypeにセットします。

```
stringToType = getApplet().getParameter("stringToType");
```

また、stringToTypeの最初の文字のキーラベルをハイライト表示します。

void stopGroup()

キーラベルやメンバ変数を初期状態に戻します。

boolean keyDown(Message msg, Event evt, int key)

入力されたキーは、引数keyで渡されます。これが文字列stringToTypeのstringIndex番目の文字と一致すれば、stringIndexを1つ進め、入力キーをエコーバック(画面に表示)し、次のキーをハイライト表示します。

入力キーのエコーバックは、ユーザ入力を表示するためのラベルコンポーネント、gui.responseFieldのtext属性を設定することで行ないます。

```
String res = (String) gui.responseField.get("text");
gui.responseField.set("text", res + String.valueOf((char)key));
```

またハイライト表示には、次に説明するguideNextKey()メソッドを呼び出します。

void guideNextKey()

文字列stringToTypeのstringIndex番目の文字をkeyTableから探し、対応するキーラベルをハイライト表示にします。また、これ以前にハイライト表示されていたキーラベルは、元に戻します。

[15] スケルトンとして生成されるメソッドのなかに、mouseDownやkeyDownなどは入っていません。これらはhandleEventメソッドの下請けとなっているためです。しかし、これらのメソッドを追加すれば、既定の処理を置き換えることができます。

キーをハイライト表示するためには、キーラベル(Visual Javaで配列したラベルコンポーネント)のbackground属性を水色(cyan)に設定します。逆に、元に戻すには白(white)に設定します。

```
keyEntry.keyTop.set("background", Color.cyan);
```

それでは、実際にプログラムを編集してみましょう。JWSにはJavaプログラムを編集するためのエディタが用意されていますので、ここではそれを使うことにします。JWSのツールバーにあるアイコンをクリックしてください。図10のようなSource Editorが表示されるはずです。

JavaKey.javaのコードは付録につけておきますので、必要なところだけ追加してください。

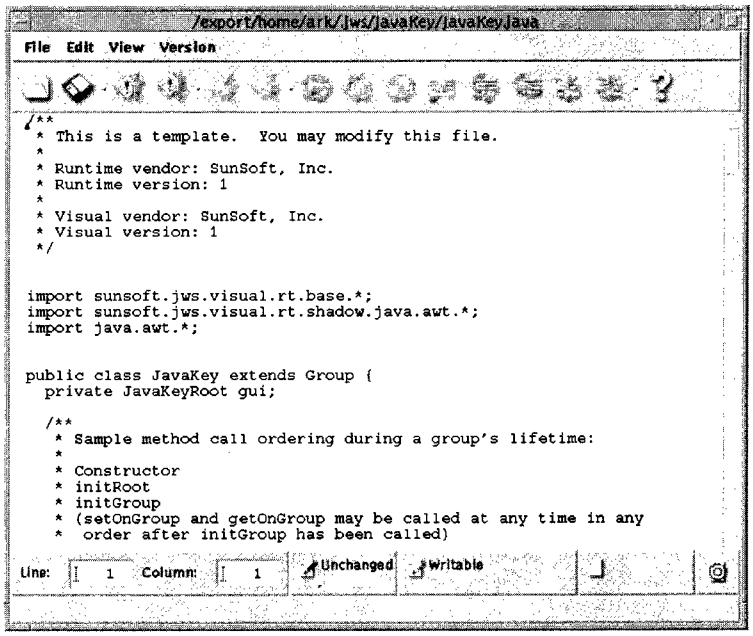


図 10: Source Editor

4.4 コンパイルと実行

Javaで記述したプログラムは、3.2節でそうしたように、実行する前にコンパイルする必要があります。ただしJWSでは、Javaコンパイラ(javac)を直接呼び出す代わりに、ボタンひとつでコンパイルできるようになっています。まず、JWSのツールバーからBuild Managerを選んでください。そしてBuild Managerのツールバーにあるアイコンをクリックすると、コンパイルが開始されます。ページの空白部にjavacを呼び出している様子が表示されるはずです。しばらくして、'Done'という表示が出れば成功です。もし画面にエラーが表示されたら、Source Editorに戻ってプログラムをもう一度見直してください。

最後に、JavaKeyに渡す文字列(タッチタイピングの練習台となる文字列)を設定しましょう。ツールバーからProject Managerを選び、表示されたページにある'Run'というタブをクリックしてください。これは、アプレット実行時の環境を設定するページになっています。ここに、アプレット実行時パラメータを設定するフィールド、Name:とValue:がありますから、それぞれ次のように入力して'Add...'ボタンを押してください。

Name:	stringToType
Value:	A quick brown fox jumped over the lazy dogs.

このページで設定した内容が、アプレットを呼び出すHTMLタグとして生成されます。この例では次のようなHTMLタグが生成され、JavaKey.tmp.html ファイルに書き出されます。

```
<applet
   name="JavaKey"
   code="JavaKeyMain.class"
   codebase="/export/home/ark/.jws/JavaKey/"
   width="433"
   height="311"
   align="Top"
   alt="If you had a java-enabled browser, you would see an applet here."
>
<param name="stringToType" value="A quick brown fox jumped over the lazy dogs.">
     <hr>If your browser recognized the applet tag,
      you would see an applet here.<hr>
</applet>
```

さあ、いよいよJavaKeyを実行するときがやってきました。JWSのツールバーから、Project Tester  を選んでください。JavaKeyが表示され、期待通りに動きましたか?キーボードが表示されているのにキー入力ができないという場合は、一旦アプレット上のどこかをクリックしてみてください。

どうしても動かないという人のために、筆者の作ったJavaKeyを、以下のURLからダウンロードできるようにしています。どうぞご利用ください。

http://falcon.center.osakafu-u.ac.jp/~ark/JavaKey/

5 おわりに

以上、駆け足でJavaの概要とプログラミングについて解説してきました。説明が足りなかったり、わかりにくい箇所が多々あったと思いますが、少しでもJavaのことを知っていただけるきっかけになれば幸いです。

実はこうしている間にもJavaをめぐる業界の動きは活発で、昨年あたりから各社がネットワークコンピュータ(NC)という新しいコンセプトに基づいたコンピュータを発表しています。これは、現在のクライアント・コンピュータの機能肥大化・複雑化に伴う、システム管理業務の増大に対するアンチテーゼとして提案されたものです。クライアント側の端末を極限まで簡略化した結果、それ自体でハードディスクを持たず、アプリケーションもデータもネットワーク上のサーバから必要なときにダウンロードして実行するしくみになっています[16]。

このほかにも、Visual Basicなどに見られるコンポーネント技術をさらに発展させたJavaBeansや、ネットワークを介したアプリケーション流通の枠組みであるCastanetなど、まだまだJavaの勢いはとどまるところを知らないようです。

最後に、Javaのキャッチコピーをお借りして本稿を締めくくりたいと思います。

Write Once, Run Anywhere, JOIN US!

[16]NC自身はファミコン並みに管理の手間がかからなくなり、一般ユーザから見れば、まさに"**Zero Administration (管理不要)**"となるそうです。

参考文献

[1] K.Arnold, J.Gosling, "The Java Programming Language", Addison Wesley, 1996.

[2] J.Gosling, B.Joy, G.Steele, "The Java Language Specification",
http://www.javasoft.com/doc/language_specification/index.html

[3] "The Java Tutorial", http://www.javasoft.com/nav/read/Tutorial/index.html

[4] B.Joy, "Javaの機能と将来像", SunWorld,Vol.6, No.2, pp.18-19, 1996.

[5] "すべてはそこから始まった – Java インサイド・ストーリー", SunWorld,Vol.6, No.2, pp.29-34, 1996.

[6] 木寺祥友, "Javaを創った人々", アスキー出版局, 1996.

[7] "特集: すべてのユーザに贈る Java再入門", Software Design, 1997年 3月号.

[8] 丸山不二夫, "Java Workshop と Group クラス", SunWorld,Vol.6, No.12, pp.112-117, 1996.

[9] 丸山不二夫, "Java Workshopのイベント処理", SunWorld,Vol.7, No.1, pp.88-93, 1997.

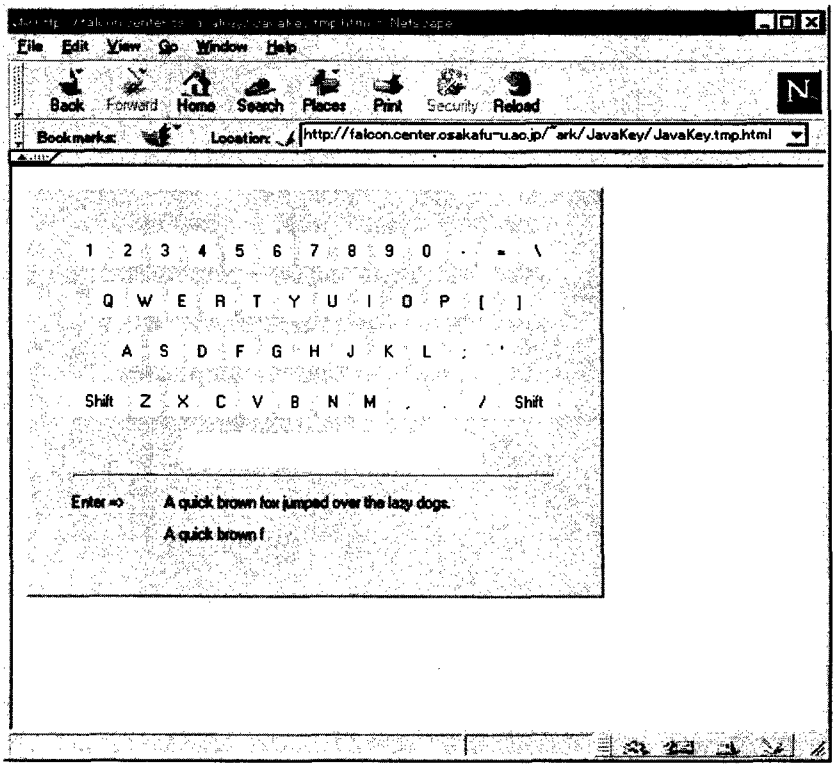


図 11: JavaKeyの実行画面

付録　JavaKey.javaプログラムリスト

```
/**
 * JWSによって自動生成されたファイルを修正したものです。
 * 紙面の都合上、コメントはカットしてあります。
 */

import sunsoft.jws.visual.rt.base.*;
import sunsoft.jws.visual.rt.shadow.java.awt.*;
import java.awt.*;

class KeyEntry {
  int           key;
  Shadow        keyTop;
  Shadow        shift;

  KeyEntry(int k, Shadow t, Shadow s) {
    key = k;
    keyTop = t;
    shift = s;
  }
}

public class JavaKey extends Group {
  private JavaKeyRoot gui;
  private KeyEntry keyTable[];
  private String stringToType;
  private int stringIndex;
  private int lastKeyIndex;

  public JavaKey() {
    addForwardedAttributes();
  }

  protected Root initRoot() {
    gui = new JavaKeyRoot(this);
    addAttributeForward(gui.getMainChild());
    return gui;
  }

  protected void initGroup() { }
  protected void showGroup() { }
  protected void hideGroup() { }
  protected void createGroup() { }
  protected void destroyGroup() { }

  protected void startGroup() {
    initKeyTable();

    stringToType = getApplet().getParameter("stringToType");
    gui.requestField.set("text", stringToType);
    stringIndex = 0;
    lastKeyIndex = -1;
    guideNextKey();
  }

  protected void stopGroup() {
    stringIndex = 0;
  }

  protected Object getOnGroup(String key) {
    return super.getOnGroup(key);
  }

  protected void setOnGroup(String key, Object value) {
    super.setOnGroup(key, value);
  }

  public boolean handleMessage(Message msg) {
    return super.handleMessage(msg);
  }
```

```
public boolean handleEvent(Message msg, Event evt) {
  return super.handleEvent(msg, evt);
}

public boolean keyDown(Message msg, Event evt, int key) {
  if (stringIndex < stringToType.length() &&
      stringToType.charAt(stringIndex) == (char)key) {
    stringIndex++;
    String res = (String) gui.responseField.get("text");
    gui.responseField.set("text", res + String.valueOf((char)key));
    guideNextKey();
    return true;
  }

  return super.keyDown(msg, evt, key);
}

void initKeyTable() {
  KeyEntry ktab[] =
    {
      new KeyEntry('1',       gui.key1,       null),
      new KeyEntry('2',       gui.key2,       null),
      new KeyEntry('3',       gui.key3,       null),
      new KeyEntry('4',       gui.key4,       null),
      new KeyEntry('5',       gui.key5,       null),
      new KeyEntry('6',       gui.key6,       null),
      new KeyEntry('7',       gui.key7,       null),
      new KeyEntry('8',       gui.key8,       null),
      new KeyEntry('9',       gui.key9,       null),
      new KeyEntry('0',       gui.key0,       null),
      new KeyEntry('-',       gui.keyMinus,   null),
      new KeyEntry('=',       gui.keyEqual,   null),
      new KeyEntry('\\',      gui.keyBSlash,  null),
      new KeyEntry('q',       gui.keyQ,       null),
      new KeyEntry('w',       gui.keyW,       null),
      new KeyEntry('e',       gui.keyE,       null),
      new KeyEntry('r',       gui.keyR,       null),
      new KeyEntry('t',       gui.keyT,       null),
      new KeyEntry('y',       gui.keyY,       null),
      new KeyEntry('u',       gui.keyU,       null),
      new KeyEntry('i',       gui.keyI,       null),
      new KeyEntry('o',       gui.keyO,       null),
      new KeyEntry('p',       gui.keyP,       null),
      new KeyEntry('[',       gui.keyLP,      null),
      new KeyEntry(']',       gui.keyRP,      null),
      new KeyEntry('a',       gui.keyA,       null),
      new KeyEntry('s',       gui.keyS,       null),
      new KeyEntry('d',       gui.keyD,       null),
      new KeyEntry('f',       gui.keyF,       null),
      new KeyEntry('g',       gui.keyG,       null),
      new KeyEntry('h',       gui.keyH,       null),
      new KeyEntry('j',       gui.keyJ,       null),
      new KeyEntry('k',       gui.keyK,       null),
      new KeyEntry('l',       gui.keyL,       null),
      new KeyEntry(';',       gui.keySColon,  null),
      new KeyEntry('\'',      gui.keyQuote,   null),
      new KeyEntry('z',       gui.keyZ,       null),
      new KeyEntry('x',       gui.keyX,       null),
      new KeyEntry('c',       gui.keyC,       null),
      new KeyEntry('v',       gui.keyV,       null),
      new KeyEntry('b',       gui.keyB,       null),
      new KeyEntry('n',       gui.keyN,       null),
      new KeyEntry('m',       gui.keyM,       null),
      new KeyEntry(',',       gui.keyComma,   null),
      new KeyEntry('.',       gui.keyPeriod,  null),
      new KeyEntry('/',       gui.keyBSlash,  null),
      new KeyEntry(' ',       gui.keySpace,   null),
      new KeyEntry('!',       gui.key1,       gui.keyRShift),
      new KeyEntry('@',       gui.key2,       gui.keyRShift),
      new KeyEntry('#',       gui.key3,       gui.keyRShift),
      new KeyEntry('$',       gui.key4,       gui.keyRShift),
      new KeyEntry('%',       gui.key5,       gui.keyRShift),
      new KeyEntry('^',       gui.key6,       gui.keyRShift),
      new KeyEntry('&',       gui.key7,       gui.keyLShift),
      new KeyEntry('*',       gui.key8,       gui.keyLShift),
      new KeyEntry('(',       gui.key9,       gui.keyLShift),
      new KeyEntry(')',       gui.key0,       gui.keyLShift),
```

```
    new KeyEntry('_',       gui.keyMinus,   gui.keyLShift),
    new KeyEntry('+',       gui.keyEqual,   gui.keyLShift),
    new KeyEntry('|',       gui.keyBSlash,  gui.keyLShift),
    new KeyEntry('Q',       gui.keyQ,       gui.keyRShift),
    new KeyEntry('W',       gui.keyW,       gui.keyRShift),
    new KeyEntry('E',       gui.keyE,       gui.keyRShift),
    new KeyEntry('R',       gui.keyR,       gui.keyRShift),
    new KeyEntry('T',       gui.keyT,       gui.keyRShift),
    new KeyEntry('Y',       gui.keyY,       gui.keyLShift),
    new KeyEntry('U',       gui.keyU,       gui.keyLShift),
    new KeyEntry('I',       gui.keyI,       gui.keyLShift),
    new KeyEntry('O',       gui.keyO,       gui.keyLShift),
    new KeyEntry('P',       gui.keyP,       gui.keyLShift),
    new KeyEntry('{',       gui.keyLP,      gui.keyLShift),
    new KeyEntry('}',       gui.keyRP,      gui.keyLShift),
    new KeyEntry('A',       gui.keyA,       gui.keyRShift),
    new KeyEntry('S',       gui.keyS,       gui.keyRShift),
    new KeyEntry('D',       gui.keyD,       gui.keyRShift),
    new KeyEntry('F',       gui.keyF,       gui.keyRShift),
    new KeyEntry('G',       gui.keyG,       gui.keyRShift),
    new KeyEntry('H',       gui.keyH,       gui.keyLShift),
    new KeyEntry('J',       gui.keyJ,       gui.keyLShift),
    new KeyEntry('K',       gui.keyK,       gui.keyLShift),
    new KeyEntry('L',       gui.keyL,       gui.keyLShift),
    new KeyEntry(':',       gui.keySColon,  gui.keyLShift),
    new KeyEntry('\"',      gui.keyQuote,   gui.keyLShift),
    new KeyEntry('A',       gui.keyZ,       gui.keyRShift),
    new KeyEntry('X',       gui.keyX,       gui.keyRShift),
    new KeyEntry('C',       gui.keyC,       gui.keyRShift),
    new KeyEntry('V',       gui.keyV,       gui.keyRShift),
    new KeyEntry('B',       gui.keyB,       gui.keyRShift),
    new KeyEntry('N',       gui.keyN,       gui.keyLShift),
    new KeyEntry('M',       gui.keyM,       gui.keyLShift),
    new KeyEntry('<',       gui.keyComma,   gui.keyLShift),
    new KeyEntry('>',       gui.keyPeriod,  gui.keyLShift),
    new KeyEntry('?',       gui.keyBSlash,  gui.keyLShift),
  };

keyTable = ktab;
```

```
  }

  int searchKey(int key) {
    for (int i = 0; i < keyTable.length; i++)
      if (keyTable[i].key == key)
        return i;
    return -1;
  }

  void hilightKeyTop(KeyEntry keyEntry, boolean hilight)
  {
    if (hilight) {
      keyEntry.keyTop.set("background", Color.cyan);
      if (keyEntry.shift != null)
        keyEntry.shift.set("background", Color.cyan);
    } else {
      keyEntry.keyTop.set("background", Color.white);
      if (keyEntry.shift != null)
        keyEntry.shift.set("background", Color.white);
    }
  }

  void guideNextKey() {
    if (stringIndex < stringToType.length()) {
      int index = searchKey((int)stringToType.charAt(stringIndex));
      if (index != -1)
      {
        if (lastKeyIndex != -1)
          hilightKeyTop(keyTable[lastKeyIndex], false);
        hilightKeyTop(keyTable[index], true);
        lastKeyIndex = index;
      }
    }
  }
}
```