ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1997
Seria: INFORMATYKA z. 33 Nr kol. 1377

Paweł GONERA
Politechnika Śląska, Instytut Informatyki

NARZĘDZIA DLA JĘZYKA JAVA

Streszczenie. W artykule przedstawiono środowiska programowania w języku Java oferowane przez komercyjnych dostawców oraz dostępne jako shareware. Przeprowadzono test wydajności różnych platform dla Javy. Przedstawiono także, w jaki sposób użyto kompilatora JIT do przyspieszania programów napisanych w Javie.

JAVA TOOLS

Summary. This paper presents integrated development environments for Java, offered by commercial suppliers and available as shareware. In paper we show efficiency tests done for various Java platforms. Also we show how JIT compiler increases the execution speed of programs written in Java.

1. Wstęp

Od pewnego czasu obserwujemy szybki wzrost popularności sieci rozległych, postrzeganych jako medium transportowe dla usługi WWW, czyli World Wide Web. Łatwy system nawigacji w sieci poprzez system połączeń hypertekstowych pomiędzy dokumentami umieszczonymi na różnych serwerach przyczynił się do tego, że z sieci Internet korzysta wielu użytkowników bez przygotowania informatycznego. Użytkownicy ci korzystają z sieci nie tylko dla celów naukowych czy biznesowych, ale także dla zabawy. Z tego powodu strony WWW stają się przepełnione grafiką, pojawiła się również muzyka oraz sekwencje wideo. Jednak stronom zapisanym w języku HTML brakuje możliwości pełnej interakcji z użytkownikiem. Jako rozszerzenie języka HTML opisującego stronę WWW powstał standard CGI (ang. Common Gateway Interface). Pozwala on na uruchomienie przez

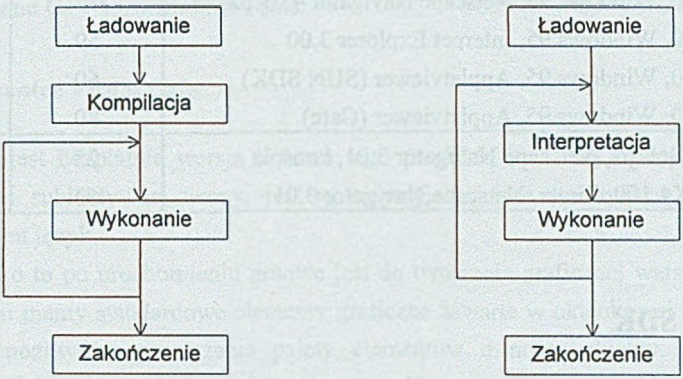
użytkownika oglądającego stronę WWW programu zapisanego na serwerze i wykonanie go. Jako wynik użytkownik dostaje kolejną stronę zawierającą efekt działania programu. Przy prostych problemach jest to doskonałe narzędzie, lecz w przypadku złożonych problemów (np. obsługa wysyłkowego sklepu) przestaje wystarczać. Jako rozszerzenie funkcjonalności stron o możliwość dowolnej interakcji z użytkownikiem firma Sun zaproponowała język Java.

Język Java nie powstał specjalnie jako „język internetowy". Został zaprojektowany jako język przenośny pomiędzy różnymi platformami sprzętowymi i programowymi na poziomie kodu źródłowego i skompilowanego. Jest to język zorientowany obiektowo o składni opartej na C++, lecz dużo prostszy. Najważniejszymi uproszczeniami są: jednostopniowe dziedziczenie, brak szablonów, brak typu wskaźnikowego. Mimo uproszczeń Java jest językiem nowoczesnym i dającym programiście duże możliwości. Posiada np. obsługę wyjątków, wielowątkowość, klasy wysokiego poziomu obsługujące połączenia sieciowe, wyświetlanie grafiki i odtwarzanie dźwięku. Dodano do tego języka elementy ułatwiające programowanie: brak konieczności usuwania tworzonych dynamicznie obiektów, zajmuje się tym tzw. **garbage collector**, gdy zmienna przestanie być potrzebna (nie ma do niej żadnego odwołania). Jednak najprawdopodobniej Java byłaby kolejnym eksperymentalnym językiem, gdyby nie możliwość publikowania małych programów nazwanych **applet** na stronach WWW [1]. Możliwość napisania programu, który przez sieć może być załadowany do komputera i będzie działał niezależnie od użytego systemu operacyjnego, spowodowała olbrzymie zainteresowanie tym językiem. Wielu producentów już teraz zadeklarowało chęć przepisania swoich produktów w Javie, ponadto rozwinęła się koncepcja tzw. komputera sieciowego (Network Computer, komputer bez dysku twardego ładujący system operacyjny i programy z serwera sieciowego), w którym oprogramowanie zawarte w pamięci ROM może uruchamiać programy napisane w Javie (producenci obiecują wydajność porównywalną do komputera klasy Pentium 200 MHz). Kolejnym powodem niezwykłej popularności Javy jest to, że firma Sun udostępnia bezpłatnie podstawowy zestaw programów pozwalający kompilować i uruchamiać programy w tym języku.

2. Interpretery i kompilatory Javy

Kosztem wieloplatformowości kodu Javy jest to, że musi on być kompilowany nie do kodu maszynowego, lecz do kodu wynikowego o specyficznej postaci tzw. kodu bajtowego. Kod ten jest zbliżony do p-kodu zaproponowanego przez N. Wirtha dla maszyny wirtualnej języka Pascal. Programy napisane w języku Java są również wykonywane przez maszynę wirtualną. Maszyna wirtualna Javy dokonuje interpretacji kodu, co jest powodem znacznego spowolnienia pracy programów w stosunku do programów napisanych w językach, których

kompilator generuje kod maszynowy (np. C, Pascal). Aby ograniczyć wpływ interpretacji kodu pośredniego, stosuje się pewien rodzaj kompilatora, który kompiluje kod bajtowy do kodu maszynowego po załadowaniu programu napisanego w Javie z sieci do komputera, na



Rys. 1. Sposób wykonania programu z kompilatorem i bez niego
Fig. 1. Program execution with and without compiler

którym jest wykonywany (patrz rys. 1).

Kompilator taki jest nazywany JIT – „Just in time compiler". Zastosowanie tego kompilatora daje efekt uboczny w postaci dodatkowego opóźnienia po załadowaniu kodu programu, ponieważ jest wtedy przeprowadzana kompilacja. Po zakończeniu kompilacji zwiększenie szybkości działania programu powinno wynagrodzić zwłokę przy jego uruchamianiu.

Aby porównać skuteczność działania kompilatorów JIT różnych producentów, napisano dwa małe applety, które wykonują dużą ilość obliczeń trygonometrycznych. Są to: animacja sześcianu obracającego się wokół jednej z osi symetrii oraz obliczanie sumy wartości wyrażenia w zadanym przedziale. Pierwszy z nich dodatkowo wykonuje operacje graficzne w celu sprawdzenia, jak wpływają na wydajność kompilatora JIT. Wynik pierwszego z nich to liczba klatek na sekundę (im więcej, tym lepiej), wynik drugiego to czas w sekundach (im mniej, tym lepiej). Sprawdzono kilka różnych programów na różnych komputerach i systemach operacyjnych. Wyniki zawiera tabela 1.

Do wyników należy dodać kilka słów komentarza. Netscape Navigator (do wersji 4.01 b3) ma najstarszy kompilator JIT (App Accelerator firmy Borland) i osiąga gorsze wyniki niż appletviewer firmy Sun, który kompilatora nie posiada. Test „Sześcian" pokazuje, że intensywne operacje graficzne zmniejszają różnice szybkości implementacji.

Tabela 1
Porównanie wydajności Javy w różnych środowiskach sprzętowych i programowych

Środowisko testowe	Sześcian [ramki/s]	Suma [s]
Pentium 100, Windows 95, Netscape Navigator 4.01 b2	57	10,06
Pentium 100, Windows 95, Internet Explorer 3.00	59	6,81
Pentium 100, Windows 95, Appletviewer (SUN SDK)	60	4,1
Pentium 100, Windows 95, Appletviewer (Cafe)	80	1,26
Sun Sparc, Solaris, Netscape Navigator 3.01, konsola	25	41,62
Intel 486DX4 100, Linux, Netscape Navigator 3.01	52	17,06

3. Java SDK

W skład pakietu dostarczanego przez firmę SUN wchodzą: kompilator *javac*, interpreter do uruchamiania samodzielnych programów *java*, narzędzie do uruchamiania programów wchodzących w skład stron WWW (appletów) *appletviewer* oraz debugger *jdb*. Kompilator sterowany jest ciągiem poleceń podawanych w linii komend. W przypadku projektu, w którego skład wchodzi wiele klas, a w Javie każda z nich zapisana jest w innym pliku, kompilacja projektu komplikuje się. Konieczne jest wtedy pisanie plików sterujących programem **make**. Nie ma, niestety, w pakiecie narzędzi wspomagających ten proces. Nienajlepsza jest szybkość pracy kompilatora, należy to jednak tłumaczyć tym, że cały pakiet dostarczany jest na kilka platform sprzętowych i programowych, więc prawdopodobnie nie był on optymalizowany na żadną z nich. Zastrzeżenia można mieć do debuggera. Przypomina on program *debug* dostarczany wraz z systemem MS-DOS we wczesnych latach osiemdziesiątych. W dobie systemów operacyjnych z graficznym środowiskiem pracy wydaje się do nich nie pasować. Zupełnie inaczej wygląda na tym tle *appletviewer*. Jest to bardzo proste narzędzie pozwalające uruchomić applety. Może on restartować, zatrzymać lub przeładować applet. Prędkość pracy, czyli jakość interpretera, jest dosyć dobra, nie dotrzymuje kroku środowiskom z nowymi kompilatorami JIT, lecz jest szybszy niż np. Netscape Navigator (tabela 1).

4. Rozwiązania komercyjne

Przedstawione tutaj zostaną środowiska programowania trzech dużych firm dostarczających kompilatory dla różnych języków. Będą to Symantec Visual Cafe, Microsoft J++ oraz Borland C++ 5.0 with Java Tools.

4.1. Symantec Visual Cafe

Dostępna jest bezpłatnie wersja testowa tego produktu i na niej oparty jest ten opis. Według mojej subiektywnej oceny, jest to najprzyjemniejsze środowisko Javy dla osób poznających ten język.

Środowisko to po uruchomieniu gotowe jest do tworzenia graficznej warstwy programu. Do dyspozycji mamy standardowe elementy graficzne zawarte w okienkowej bibliotece Javy AWT oraz możliwość rozszerzania palety elementów o nowe obiekty. Jako przykład producent dodał proste klasy demonstracyjne: klasa realizująca drgający tekst, efekt fajerwerków i bardziej użyteczna klasa realizująca okienko zawijające długi napis na wiele linii.

Po zaprojektowaniu wyglądu ekranu należy oprogramować umieszczone elementy sterujące. Istnieją tu co najmniej dwie drogi. Można użyć tzw. Interaction Wizard, jest to narzędzie służące do łączenia elementów sterujących ze sobą. Jego działanie polega na wybraniu elementów współdziałających ze sobą i określeniu zdarzeń oraz odpowiadających im metod. Drugim sposobem jest napisanie metod obsługi zdarzeń samemu. Cafe pomaga nam w tym. Jeżeli program ma zareagować na naciśnięcie przycisku „Start", to na oknie appletu zaznaczamy ten przycisk, następnie w okienku właściwości wybieramy z listy zdarzenie *OnClick* i automatycznie przechodzimy do edytora do szablonu metody wirtualnej OnClick. Automatyzacja postępuje jednak nieco dalej. Modyfikacji ulega (lub zostaje utworzona) metoda *HandleEvent* odpowiedzialna za wywoływanie metod w zależności od rodzaju zdarzenia. Ta możliwość wydaje mi się szczególnie interesująca, ponieważ daje spore możliwości automatyzacji pisania standardowych fragmentów programu, jednocześnie nie dając poczucia „nachalności" środowiska programowania. W rozdziale 6 zamieszczony jest kod źródłowy appletu „Suma" napisanego w środowisku Visual Cafe. Linie automatycznie wygenerowane przez program zostały oznaczone komentarzem „//AUTOMAT".

Po utworzeniu programu następuje proces jego uruchamiania. W środowisku Visual Cafe mamy typowy debugger. Daje on możliwość podglądania zmiennych, ustawiania pułapek oraz pracy krokowej. W trakcie uruchamiania appletów współdziała z programem appletviewer. sterując procesem wykonywania appletu.

4.2. Microsoft J++

Produkt ten jest oparty na standardowym dla firmy środowisku programowania nazywanym Visual Studio. Firma Microsoft nie użyła SDK Javy, lecz napisała własną implementację, zgodną z pierwowzorem. J++ jest dobrą platformą programistyczną zaopatrzoną w dobry edytor, szybki kompilator oraz zintegrowany debugger pozwalający na łatwe śledzenie stanu zmiennych, pracę krokową oraz umożliwiający zakładanie pułapek. Jako platforma do uruchamiania appletów służy przeglądarka Internet Explorer. Podczas uruchamiania appletów debugger potrafi sterować procesem wykonywania w przeglądarce, umożliwiając efektywne usuwanie błędów. Do Internet Explorera dołączony jest kompilator JIT, dający bardzo szybki kod wynikowy (patrz tabela 1).

4.3. Borland C++ 5.01 z Java tools

Firma Borland zdecydowała się na dodanie do pakietu C++ możliwości tworzenia programów w języku Java. Została dodana obsługa pakietu SDK firmy Sun. Sposób, w jaki zostało to zrealizowane, pozostawia wiele do życzenia. Pierwszym z zarzutów jest to, iż aby pisać w Javie, należy zakupić kompilator C++, nie można rozdzielić pakietu na część dotyczącą Javy i C++. Nie ma w pakiecie debuggera do uruchamiania appletów, można korzystać jedynie z możliwości śledzenia programu zawartego w środowisku uruchomieniowym, czyli ograniczono się tylko do programów wolno stojących. Poważną wadą jest brak jakiegokolwiek narzędzia do projektowania pośrednictwa użytkownika. Zaletą jest możliwość korzystania z zaawansowanego środowiska zintegrowanego. Zapewnia ono tworzenie projektu, edytor ma możliwość podświetlania składni, co eliminuje sporo błędów syntaktycznych. Również debugger jest bardzo dobrym narzędziem, pod warunkiem, że piszemy programy wolno stojące.

Narzędzie to będzie użyteczne, jeżeli posiadamy już pakiet i dokupimy do niego rozszerzenie o obsługę Javy. Nie trzeba wtedy zmieniać swoich przyzwyczajeń. Mimo to do niektórych problemów jest ono niewystarczające.

5. Rozwiązania shareware

Amatorzy stworzyli wiele narzędzi programistycznych dostępnych jako shareware. Na początku były to edytory, które posiadały dodatkowe narzędzia typu make, później pojawiły się bardziej zaawansowane środowiska programistyczne wyposażone w dobre edytory z podświetlaniem składni, okienko projektu, rozpoznawanie miejsca wystąpienia błędu kompilacji na podstawie wyniku działania kompilatora i inne udogodnienia. Cechą wspólną

jest jednak brak zintegrowanego debuggera i korzystanie z kompilatora pochodzącego z SDK firmy Sun. Oprócz programów przeznaczonych dla programistów pojawiła się duża ilość konfigurowalnych appletów przeznaczonych dla osób, które nie umieją programować w Javie, a chcą umieścić coś ładnego na swojej stronie. Tą grupą programów nie będę się zajmował. Poniżej przedstawię cechy typowego środowiska zintegrowanego Javy dostępnego jako shareware.

5.1. Javadraw

Jest to ograniczone środowisko programistyczne zaopatrzone w edytor z podświetlaniem składni i możliwość tworzenia projektu. Nie ma ono własnego kompilatora, lecz korzysta z kompilatora firmy Sun. W przypadku wystąpienia błędu kompilacji rozpoznawane jest miejsce jego wystąpienia. Po poprawnej kompilacji Javadraw oferuje nam możliwość uruchomienia gotowego programu. W przypadku appletu mamy możliwość wybrania programu (np. appletviewer), za którego pomocą będziemy go oglądali.

Oprócz tych podstawowych funkcji Javadraw oferuje zaawansowane funkcje edytora, takie jak wyszukiwanie metod oraz tzw. Ghostwriter. Wyszukiwanie metod polega na tym, że w okienku z listą wszystkich metod zdefiniowanych w projekcie wybieramy interesującą nas metodę i otwiera się okno edytora z poszukiwaną metodą. Ghostwriter to proste narzędzie do tworzenia szablonów struktur językowych Javy, np. jeżeli wybierzemy konstrukcję „if ()", to do edytora wpisze się konstrukcja if-else, wraz z nawiasami. Jednak narzędzie takie należy traktować jako ciekawostkę. Bardziej użyteczne jest tworzenie szablonu klasy pochodnej po klasie Applet, jeżeli utworzymy nowy projekt. Tworzony jest plik źródłowy z definicją klasy o nazwie zgodnej z nazwą projektu i utworzonymi metodami, które należy przedefiniować. Nie jest obowiązkowe definiowanie wszystkich proponowanych przez Javadraw metod.

Javadraw jest dosyć wygodnym środowiskiem pracy dla niewymagającego programisty. Brakuje tutaj jedynie debuggera, aby zarekomendować ten program jako „niskobudżetowe środowisko programistyczne dla Javy". Autorem tego programu jest firma SFS Software z Schmalkalden w Niemczech.

6. Dodatek

Poniżej przedstawiony został tekst źródłowy appletu testującego „Suma" używanego do testowania środowisk Javy w rozdziale 2. Komentarzem „//AUTOMAT" oznaczono linie wygenerowane przez środowisko Visual Cafe, w którym powstał ten applet.

```
import java.awt.*;                                    //AUTOMAT
```

```
import java.applet.*;                                                    //AUTOMAT
import java.util.*;                                                      //AUTOMAT

public class wydaj extends Applet {
  public boolean handleEvent(Event event) {                              //AUTOMAT
    if (event.target==start_but && event.id==Event.ACTION_EVENT) //AUTOMAT
      startBut_Clicked(event);                                           //AUTOMAT
    return super.handleEvent(event);                                     //AUTOMAT
  }

  void startBut_Clicked(Event event) {                                   //AUTOMAT
    wynik.setText("");
    start_but.setLabel("Pracuje");
    wynik.setText(policz());
    start_but.setLabel("Start");
  }

  public String policz(){
    float wyn = 0;
    float i = -500;
    Date czas_strt = new Date();

    while (i<500){
      wyn += Math.sin( i );
      i+=0.001;
    }
    Date czas_stp = new Date();
    Date tymcz    = new Date(czas_stp.getTime() - czas_strt.getTime());
    return Float.toString((float)(tymcz.getTime())/1000);
    }

  public void init() {                                                   //AUTOMAT
    super.init();                                                        //AUTOMAT
    setLayout(null);                                                     //AUTOMAT
    addNotify();                                                         //AUTOMAT
    resize(294,153);          //AUTOMAT (na podstawie formy graficznej)
    setBackground(new Color(12632256));                                  //AUTOMAT
    start_but = new java.awt.Button("Start");                            //AUTOMAT
    start_but.reshape(51,105,192,32);                                    //AUTOMAT
    start_but.setFont(new Font("Dialog", Font.BOLD, 12));                //AUTOMAT
    add(start_but);                                                      //AUTOMAT
    label1 = new java.awt.Label("Czas");                                 //AUTOMAT
    label1.reshape(35,45,42,15);                                         //AUTOMAT
    label1.setFont(new Font("Dialog", Font.PLAIN, 12));                  //AUTOMAT
    label1.setBackground(new Color(12632256));                           //AUTOMAT
    add(label1);                                                         //AUTOMAT
    wynik = new java.awt.Label("",Label.RIGHT);                          //AUTOMAT
    wynik.reshape(84,45,91,15);                                          //AUTOMAT
    wynik.setFont(new Font("Courier", Font.PLAIN, 12));                  //AUTOMAT
    add(wynik);                                                          //AUTOMAT
  }

  //{{DECLARE_CONTROLS
  java.awt.Button start_but;                                             //AUTOMAT
  java.awt.Label label1;                                                 //AUTOMAT
  java.awt.Label wynik;                                                  //AUTOMAT
  //}}
}
```

LITERATURA

[1] Anuff E.: The Java Sourcebook. J. Wiley & Sons, Inc, New York 1996

[2] Morrison M.: Java Unleashed. Sams Publishing, Indianapolis 1996.

[3] Van Hoff A, Shaio S., Starbuck O.: Java, Helion, Gliwice 1996.

Recenzent: Dr inż. Maciej Bargielski

Wpłynęło do Redakcji 7 maja 1997 r.

Abstract

This paper presents various development tools for Java. In first chapter I present reasons why Java become very popular „Internet programming language". Next chapter presents the way how we can increase execution speed of the programs written in Java. Figure 1 contains diagrams how the Java program is executed with and without „just in time compiler". Next in this chapter I presents results of execution speed tests, done on the several Java platforms (Table 1). Last part of this paper presents some programming environments. The platforms are: Visual Cafe, J++, Borland C++ with Java Tools. I also presents simple programming environment JavaDraw available as shareware.