W większości podręczników dotyczących programowania pierwsza lekcja stanowi materiał lekki, łatwy i przyjemny, tak aby nie przepłoszyć czytelnika. Nasz podręcznik nie różni się pod tym względem od innych, tyle tylko, że intencją Autorów jest, aby również następne były może nie tyle lekkie i łatwe, co przyjemne i zajmujące.

Materiał pierwszej lekcji obejmuje krótkie omówienie idei programowania obiektowego i, oczywiście, przedstawienie jego zalet (oraz nielicznych wad). Niewątpliwie najbardziej znaną, popularną, i co również ważne - bezpłatną platformą programowania obiektowego jest **Java** stworzona w firmie Sun Microsystems (przejętej obecnie przez Oracle Corporation). Należy jednak zdawać sobie sprawę, że chociaż Java jest najlepsza ☺, to istnieją również inne platformy. Do bardziej znanych (także bezpłatnych) należy np. FreePascal http://www.freepascal.org/. W trakcie niniejszej lekcji dowiecie się, gdzie znaleźć i w jaki sposób zainstalować (i odinstalować) na Waszym komputerze pakiet Javy. Jego instalacja jest niezbędna do dalszego pełnego korzystania z podręcznika.

Jak to zwykle bywa - po pierwszej lekcji nie będzie zadań domowych (poza zadaniem zainstalowania Javy). A teraz zapraszamy do pracy!

1. Zalety (i wady) programowania obiektowego

Zbudowanie komputera, takiego jak ten, za pomocą którego czytasz ten podręcznik, wymagało całego sztabu ludzi, wyposażonych w wiedzę i najnowszą technologię, pracujących przez wiele lat nad jego poszczególnymi elementami, a następnie połączenia wyników ich pracy w jeden, działający system, nieustannie rozwijany i udoskonalany. Pełne zrozumienie istoty działania Twojego komputera, niezbędne do jego zaprojektowania, wymaga bardzo szerokiej, specjalistycznej wiedzy z zakresu fizyki ciała stałego, technologii półprzewodnikowych, układów logicznych, przetwarzania sygnałów i wielu innych dziedzin. Przytłaczająca większość ludzi nie dysponuje tak wszechstronną wiedzą, jednak bardzo wielu z nich potrafi się biegle posługiwać komputerem, a prawie każdy potrafi go zmusić do działania, naciskając przycisk włącznika. Ba - można się spodziewać, że większość z Was potrafi samodzielnie poskładać i uruchomić komputer z podstawowych "klocków", do których należy obudowa, zasilacz, karta graficzna, stacje dysków, modem, monitor itp. połączone ze sobą w odpowiedni sposób. Cała **złożoność** systemu jest nieistotna - **uogólniamy** go do zjawiska, nazywanego komputerem, które pełni rolę **czarnej skrzynki**, zachowującej się zasadniczo w przewidywalny (mniej lub bardziej ☺) sposób. Mówiąc bardziej naukowo - nie rozumiejąc, lub nie potrzebując zrozumieć całej skomplikowanej techniki - tworzymy **model abstrakcyjny urządzenia**, który pozwala nam obsługiwać komputer nie zawracając sobie głowy szczegółami typu działanie tranzystorów, czy układów pamięci.

(1.1) Ogólna idea

Taki, intuicyjny zresztą, sposób pojmowania komputera jest w istocie odzwierciedleniem idei programowania zorientowanego obiektowo.

Mając za zadanie zbudowanie komputera, rozejrzymy się przede wszystkim za jego elementami składowymi - obiektami klas składowych komputera, do których zaliczymy zasilacz, stacje dysków etc. Następnie połączymy je ze sobą i przy odrobinie szczęścia uzyskamy obiekt klasy komputer. Jeszcze prostszą metodą jest pójście do sklepu komputerowego, gdzie półki uginają się pod wielką ilością obiektów klasy komputer, i kupienie sobie jednego z nich. **No, proszę - nic o tym nie wiedząc działamy jak programista obiektowy.**

Programista proceduralny, mając za zadanie zbudowanie komputera, prawdopodobnie rozpocząłby od zaprojektowania tranzystora, by po przejściu przez etapy układów scalonych różnej skali integracji, technologii płytek drukowanych itp. doprowadzić do skonstruowania procesora. Teraz pozostaje tylko skonstruować monitor, pamięci magnetyczne, rozwiązać problem zasilania i chłodzenia, dodać urządzenia I/O i już prawie gotowe ☺. Oczywiście po zakończeniu pracy będzie wiedział dużo więcej niż my, ale za to z naszego komputera można skorzystać jakieś 40 lat wcześniej.

Głównym założeniem programowania obiektowego jest, że program komputerowy jest zbiorem oddzielnych obiektów, połączonych w jedną całość, komunikujących się pomiędzy sobą i wpływających na siebie nawzajem. W praktyce pisanie programu niejednokrotnie sprowadza się do twórczego wykorzystania już istniejących obiektów, stworzonych przez kogoś innego, czasami do zupełnie innego celu. Dokumentacja Javy zawiera kilka tysięcy plików, zawierających definicje gotowych do wykorzystania klas, metod, zmiennych itp. Podobnie rozbudowane są inne środowiska programowania obiektowego, jak C++ czy Pascal.

(1.2) Zalety...

Jak widać podstawową zaletą programowania obiektowego jest ograniczenie liczby elementów, nad którymi programista musi jednocześnie zapanować. Pogrupowane w klasy elementy dają się łatwo opanować, a poprzez fakt, że odzwierciedlają naturalny sposób myślenia człowieka czynią programowanie bardziej oczywistym.

Program napisany w języku obiektowym, ze względu na swą strukturę działa bezpieczniej niż program proceduralny (uszkodzenie modemu nie musi spowodować awarii całego komputera) i, z tego samego powodu, łatwiej testować jego poszczególne fragmenty.

Dodatkową, niezwykle istotną zaletą, będącą konsekwencją tworzenia bibliotek klas i mechanizmów dziedziczenia, jest możliwość wielokrotnego użycia raz napisanego kodu, połączona z łatwością modyfikacji jego fragmentów - zamiast tworzyć całą klasę od nowa można wykorzystać klasę już zdefiniowaną, zmieniając w jej obrębie pojedyncze metody, co jest idealnym rozwiązaniem dla leniwych programistów ☺.

(1.3) ...i wady

Oczywiście programowanie obiektowe nie jest bez wad. Ponieważ jest odzwierciedleniem naturalnego podejścia człowieka do rzeczywistości, to znalezienie błędu w niedziałającym kodzie programu napisanego przez bałaganiarskiego programistę przypomina próbę znalezienia dwóch jednakowych skarpet w wielkiej szafie, wypełnionej skłębioną odzieżą wszystkich domowników. W tym przypadku łatwiej kupić nową parę, czyli napisać program od nowa. Tutaj oczywiście programista proceduralny, z siłą rzeczy przejrzystym, sekwencyjnym algorytmem będzie się z nas śmiał w kułak. Aby do tego nie dopuścić należy po prostu wyrobić w sobie nawyk pisania w sposób porządny i elegancki, pamiętając o tym, że klasy, które tworzymy mogą się jeszcze niejednokrotnie przydać.

Druga wada również jest konsekwencją zalet - dostępność biblioteki rozmaitych klas jest bardzo wygodna, tyle tylko, że ta biblioteka jest ogromna - i powiększa się z każdą chwilą, tak, że nawet z dobrym indeksem rzeczowym w ręku czasami ciężko coś w niej znaleźć. Niestety, zdarza się, że tworząc aplikację obiektową znaczną część czasu spędzamy szperając w dokumentacji, a nie programując. No cóż - z tą akurat wadą trzeba się pogodzić.

2. Instalowanie Javy

W tym miejscu wyjaśnimy Ci, w jaki sposób zainstalować aplikacje niezbędne do nauki "Programowania obiektowego".

(2.1) Instalacja pakietu JAVA

Aby zainstalować pakiet JAVA należy najpierw ściągnąć program instalacyjny środowiska **JDK** ze strony internetowej http://www.oracle.com/technetwork/java/.

Po otwarciu powyższej strony należy przejść do zakładki **Downloads**, następnie wybrać kategorię **Java SE** i w dziale **Java Platform, Standard Edition** nacisnąć przycisk **Download JDK**, a potem **Download**, co - po wybraniu systemu operacyjnego działającego na naszym komputerze - pozwoli w końcu ściągnąć odpowiedni program instalacyjny. Ściągnięty program należy uruchomić i postępować zgodnie z jego wskazówkami.

Uruchomiony instalator zapyta, które składniki środowiska **JDK** mają zostać zainstalowane oraz w jakim katalogu. Naciśnięcie przycisku **Next >** spowoduje zaakceptowanie domyślnych ustawień. Dalej nastąpi pytanie o katalog, w którym ma zostać zainstalowane środowisko uruchomieniowe **JRE**. Kolejne naciśnięcie **Next >** spowoduje zaakceptowanie domyślnego katalogu i rozpoczęcie właściwego procesu instalacji. W końcu program instalacyjny poinformuje nas o poprawnym zakończeniu działania i poprosi o naciśnięcie przycisku **Finish**.

Przykładowo, w systemach rodziny **MS Windows**, w rezultacie poprawnie przeprowadzonej instalacji pakietu JAVA, w katalogu **Program Files** pojawi się podkatalog **Java** zawierający dwa podkatalogi, odpowiednio z **JDK** i **JRE**. Dodatkowo w systemie zarejestrowany zostanie nowy typ plików: **Executable Jar File**, którego podwójne kliknięcie zostanie związane z uruchomieniem programu **`javaw.exe`**.

(2.2) Instalacja dokumentacji pakietu JAVA

Aby zainstalować dokumentację pakietu JAVA należy najpierw ściągnąć spakowany plik dokumentacji ze strony internetowej http://www.oracle.com/technetwork/java/.

Po otwarciu powyższej strony należy przejść do zakładki **Downloads**, następnie wybrać kategorię **Java SE**, w dziale **Additional Resources** przejść do **Java SE Documentation** i nacisnąć przycisk **Download Zip**, co - po wybraniu języka dokumentacji oraz zaakceptowaniu warunków licencji - pozwoli ściągnąć odpowiedni plik. Uzyskany plik należy rozpakować do katalogu zawierającego środowisko **JDK**.

Przykładowo, w systemach rodziny **MS Windows**, środowisko **JDK** w wersji 6.0_20 znajduje się w katalogu **C:\Program Files\Java\jdk1.6.0_20** i tu należy umieścić podkatalog **docs** powstający w wyniku rozpakowania pliku z dokumentacją. Rozpakowana dokumentacja Javy składa się z wielu tysięcy plików umieszczonych w złożonym drzewie katalogowym.

UWAGA: W przypadku braku odpowiedniego programu rozpakowującego, testową wersję popularnego **`WinZip-a`** można ściągnąć ze strony http://www.winzip.com/.

(2.3) Ustawienia systemu ułatwiające pracę

Po pomyślnym zakończeniu instalacji najważniejsze narzędzia Javy będą zgromadzone w podkatalogu **bin** znajdującym się w katalogu zawierającym środowisko **JDK**. Oto niektóre z nich:

javac - kompilator,
java - interpreter z konsolą,
javaw - interpreter bez konsoli,
javadoc - generator dokumentacji API,
appletviewer - interpreter apletów,
jar - program do tworzenia i zarządzania archiwami JAR,
jdb - debugger.

Pisząc programy w Javie stale będziemy z nich korzystali, dlatego wygodnie będzie ustawić w systemie operacyjnym ścieżkę dostępu do tego katalogu.

Przykładowo, w systemach rodziny **MS Windows** będzie to katalog **C:\Program Files\Java\jdk1.6.0_20\bin** i pełną jego nazwę należy dodać do zmiennej środowiska **PATH**:

- w **MS Windows 95/98** zmienną **PATH** ustawiało się w pliku **AUTOEXEC.BAT**;
- w **MS Windows NT** ustawienie zmiennej **PATH** wymagało kliknięcia prawym klawiszem myszy na ikonie **Mój komputer**, przejścia do **Właściwości**, wybrania zakładki **Środowisko** i ustawienia prawidłowej wartości zmiennej **PATH**;
- w **MS Windows XP** po kliknięciu prawym klawiszem myszy na ikonie **Mój komputer** i przejściu do **Właściwości**, należy wybrać zakładkę **Zaawansowane**, nacisnąć przycisk **Zmienne środowiska** i odpowiednio wyedytować zmienną **PATH**.

Aby pliki Javy, zarówno teksty programów, jak i skompilowane klasy były łatwo rozróżnialne w katalogach, dobrze jest utworzyć dwa nowe typy plików i związać je z odpowiednimi ikonami. Katalog **download** na płycie CD-ROM zawiera przykładowe pliki ikon: Java.ico - ikona pliku z tekstem programu (rozszerzenie **java**) oraz Java-class.ico - ikona skompilowanej klasy (rozszerzenie **class**).

Przykładowo, chcąc wykorzystać te pliki w systemach rodziny **MS Windows**, należy je skopiować na dysk twardy (najlepiej do katalogu **C:\WINDOWS**), a następnie zarejestrować nowy typ plików:

- w **MS Windows 95/98** w *Menu Exploratora / Widok / Opcje folderów...* należało przejść do zakładki *Typy plików*, a następnie nacisnąć przycisk **Nowy typ…** i dokonać odpowiedniej rejestracji wraz z ustawieniem ikony;
- w **MS Windows NT** w *Menu Exploratora / Widok / Opcje...* należało przejść do zakładki *Typy plików*, a następnie nacisnąć przycisk **Nowy typ…** i dokonać odpowiedniej rejestracji wraz z ustawieniem ikony;
- w **MS Windows XP** w *Menu Exploratora / Narzędzia / Opcje folderów...* należy przejść do zakładki *Typy plików*, następnie nacisnąć przycisk **Nowy**, w pole okna dialogowego wpisać odpowiednie rozszerzenie: **java** lub **class**, nacisnąć **OK**, a następnie nacisnąć **Zaawansowane** i ustawić ikonę (przycisk **Zmień ikonę…**).

Spowodowanie, by dwukrotne kliknięcie na ikonę pliku z tekstem programu uruchamiało jego edycję wymaga dodatkowo stworzenia nowej akcji związanej z definiowanym typem plików. Należy zatem utworzyć akcję pod nazwą **Otwórz** lub **Edytuj** i definiując ją podać ścieżkę do używanego edytora tekstowego (np. systemowego Notatnika).

Rzecz jasna, możemy ułatwić sobie życie i ściągnąć z Internetu, a następnie zainstalować w naszym komputerze jeden z darmowych edytorów do pisania programów w Javie. Edytory te kolorują składnię języka oraz integrują się ze środowiskiem Javy i tworzą wygodne środowisko pracy programisty **IDE**, czyli kompilacja i uruchamianie programów następuje poprzez kliknięcie ikony na pasku w edytorze, a nie poprzez żmudne pisanie polecenia w konsoli systemu operacyjnego. Oto najpopularniejsze:

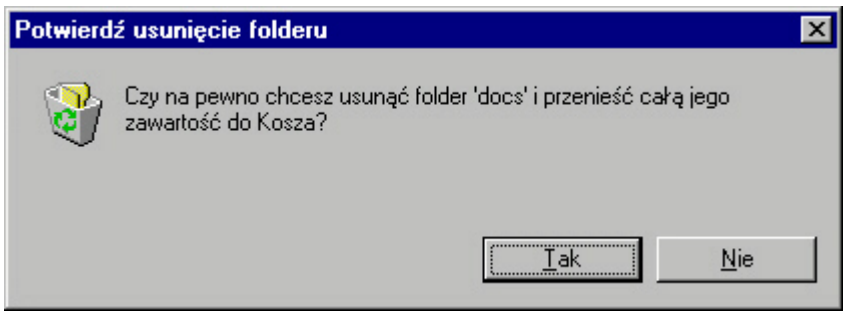
- http://netbeans.org/downloads/ - NetBeans - środowisko stworzone przez samych dostawców Javy (można je nawet ściągnąć z samą Javą);
- http://www.jcreator.com/ - JCreator - wygodne środowisko zainstalowane na komputerach,

na których będzie odbywał się egzamin z **Programowania obiektowego** ☺;
- http://www.elipse.org/ - Eclipse - w pełni profesjonalne i darmowe ☺ środowisko do tworzenia złożonych projektów przez duże zespoły programistów.

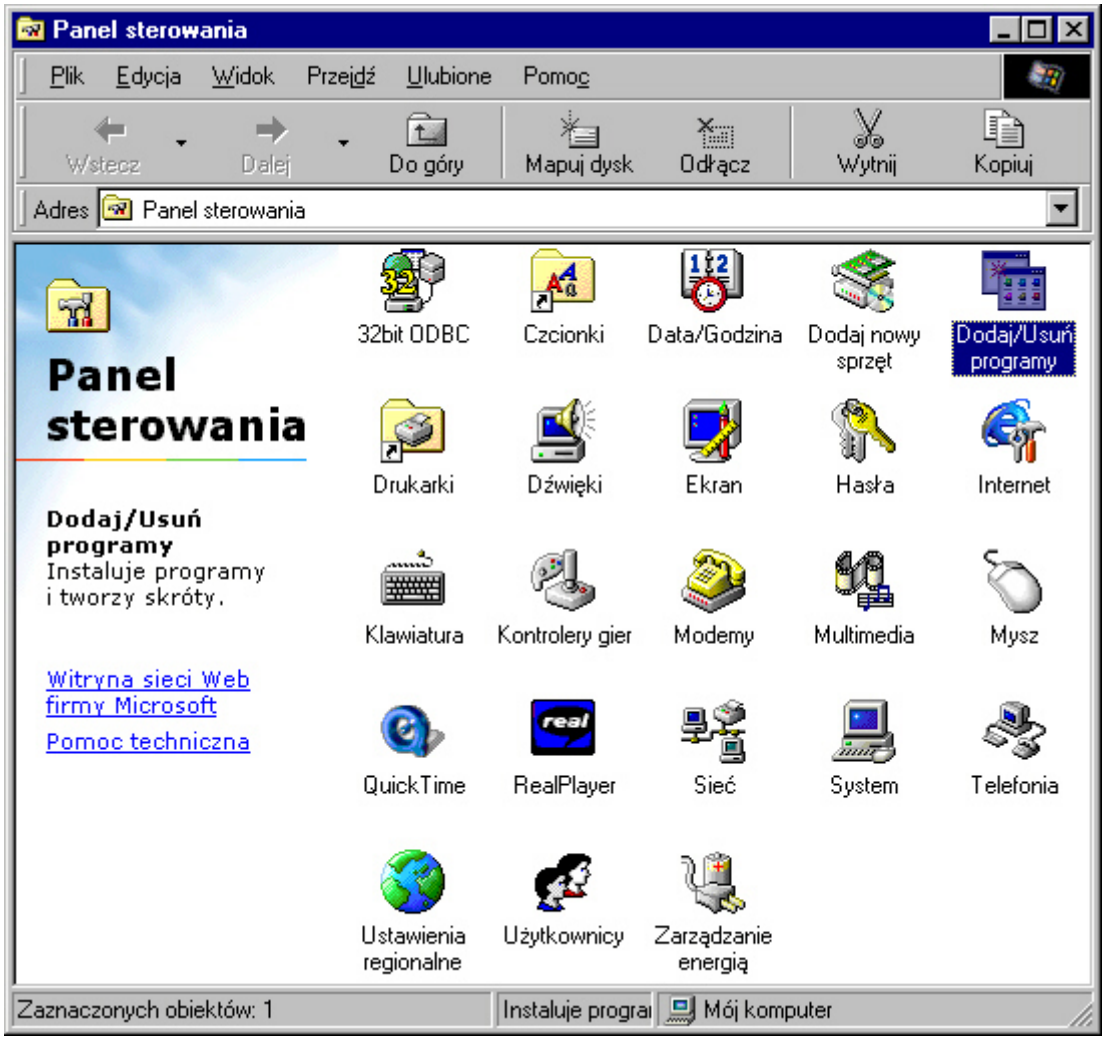
Warto pamiętać, że zainstalowanie któregoś z wyżej wymienionych środowisk spowoduje zmiany wprowadzonych wcześniej przez nas ustawień dotyczących typów plików **java** i **class**.

(2.4) Upgrade pakietu JAVA do nowszej wersji

Aby zainstalować nowszą wersję Javy, należy najpierw **ODINSTALOWAĆ** wersję istniejącą w systemie - zwykłe skasowanie katalogów zawierających narzędzia Javy może spowodować w przyszłości niestabilną pracę systemu operacyjnego! Jednakże zastrzeżenie to nie dotyczy podkatalogu **docs** zawierającego dokumentację Javy - został on przecież dograny po zainstalowaniu całego pakietu. Zatem, katalog ten trzeba skasować ręcznie - program deinstalacyjny nie jest w stanie samodzielnie go usunąć!



Po usunięciu katalogu **docs** można przystąpić do właściwej deinstalacji. W systemach rodziny **MS Windows** najpierw należy wybrać opcję *Menu Start / Ustawienia / Panel Sterowania*, a następnie uruchomić, w zależności od wersji systemu, narzędzie **Dodaj/Usuń programy** albo **Dodaj lub usuń programy**.



Z listy zainstalowanych aplikacji należy po kolei wybrać wszystkie związanie z pakietem Javy - ich nazwy zaczynają się od **Java** - a na następnie naciskając **Dodaj/Usuń**, **Zmień/Usuń** bądź **Usuń** uruchomić deinstalator.

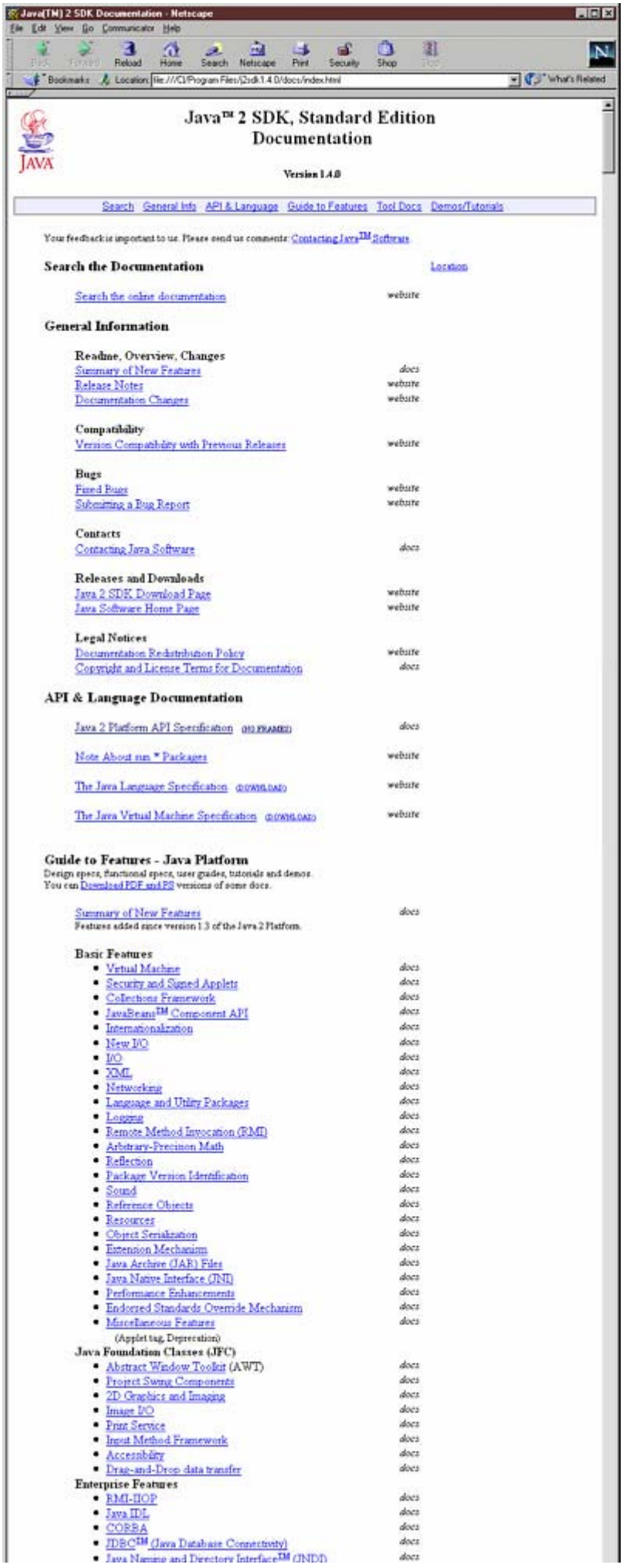
Trzeba też pamiętać o usunięciu wpisu dotyczącego **ścieżki dostępu** w pliku **AUTOEXEC.BAT** albo w ustawieniach zmiennej środowiska **PATH**.

3. Korzystanie z dokumentacji

Umiejętność korzystania z dokumentacji jest niezwykle przydatna zarówno w trakcie nauki programowania, jak i później, podczas ewentualnego tworzenia aplikacji w pracy zawodowej. Nie sposób przecież zapamiętać wszystkich dostępnych w danym języku konstrukcji i funkcji. Zresztą z wersji na wersję ulegają one (nie zawsze drobnym) modyfikacjom. Niektóre elementy uznane zostają za przestarzałe i poleca się programistom, by ich nie używali. Wprowadzane są nowe elementy, często rozwiązujące istniejące wcześnie trudne problemy. Zatem aktualna dokumentacja pozwala "być na bieżąco", szczególnie w dynamicznie rozwijanych językach, takich jak Java. Podstawy oczywiście trzeba znać, bo trudno przy pisaniu kolejnego programu uczyć się wszystkiego od początku, ale wiedza, gdzie szukać pożądanych informacji pozwala zaoszczędzić mnóstwo cennego czasu.

(3.1) Najważniejsze działy dokumentacji

Dokumentacja Javy dostarczana jest postaci biblioteki plików **HTML**. Aby móc z niej skorzystać, należy w przeglądarce otworzyć dokument `index.html` znajdujący się w podkatalogu **docs** katalogu zawierającego narzędzia **JDK** Javy. Jak można zauważyć na poniższym obrazku pokazującym przykładową dokumentację Javy **1.4.0**, lista punktów, które się na nią składają, jest przerażająco długa…



Java(TM) 2 SDK Documentation - Netscape

File Edit View Go Communicator Help

Back Forward Reload Home Search Netscape Print Security Shop Stop

Bookmarks Location: file:///C|/Program Files/j2sdk1.4.0/docs/index.html What's Related

Java™ 2 SDK, Standard Edition Documentation

Version 1.4.0

Search | General Info | API & Language | Guide to Features | Tool Docs | Demos/Tutorials

Your feedback is important to us. Please send us comments: Contacting Java™ Software

Search the Documentation — Location

Search the online documentation — website

General Information

Readme, Overview, Changes

- Summary of New Features — docs
- Release Notes — website
- Documentation Changes — website

Compatibility

- Version Compatibility with Previous Releases — website

Bugs

- Fixed Bugs — website
- Submitting a Bug Report — website

Contacts

- Contacting Java Software — docs

Releases and Downloads

- Java 2 SDK Download Page — website
- Java Software Home Page — website

Legal Notices

- Documentation Redistribution Policy — website
- Copyright and License Terms for Documentation — docs

API & Language Documentation

- Java 2 Platform API Specification (NO FRAMES) — docs
- Note About sun.* Packages — website
- The Java Language Specification (DOWNLOAD) — website
- The Java Virtual Machine Specification (DOWNLOAD) — website

Guide to Features - Java Platform

Design specs, functional specs, user guides, tutorials and demos.
You can Download PDF and PS versions of some docs.

- Summary of New Features — docs
  Features added since version 1.3 of the Java 2 Platform.

Basic Features

- Virtual Machine — docs
- Security and Signed Applets — docs
- Collections Framework — docs
- JavaBeans™ Component API — docs
- Internationalization — docs
- New I/O — docs
- I/O — docs
- XML — docs
- Networking — docs
- Language and Utility Packages — docs
- Logging — docs
- Remote Method Invocation (RMI) — docs
- Arbitrary-Precision Math — docs
- Reflection — docs
- Package Version Identification — docs
- Sound — docs
- Reference Objects — docs
- Resources — docs
- Object Serialization — docs
- Extension Mechanism — docs
- Java Archive (JAR) Files — docs
- Java Native Interface (JNI) — docs
- Performance Enhancements — docs
- Endorsed Standards Override Mechanism — docs
- Miscellaneous Features — docs
  (Applet tag, Deprecation)

Java Foundation Classes (JFC)

- Abstract Window Toolkit (AWT) — docs
- Project Swing Components — docs
- 2D Graphics and Imaging — docs
- Image I/O — docs
- Print Service — docs
- Input Method Framework — docs
- Accessibility — docs
- Drag-and-Drop data transfer — docs

Enterprise Features

- RMI-IIOP — docs
- Java IDL — docs
- CORBA — docs
- JDBC™ (Java Database Connectivity) — docs
- Java Naming and Directory Interface™ (JNDI) — docs

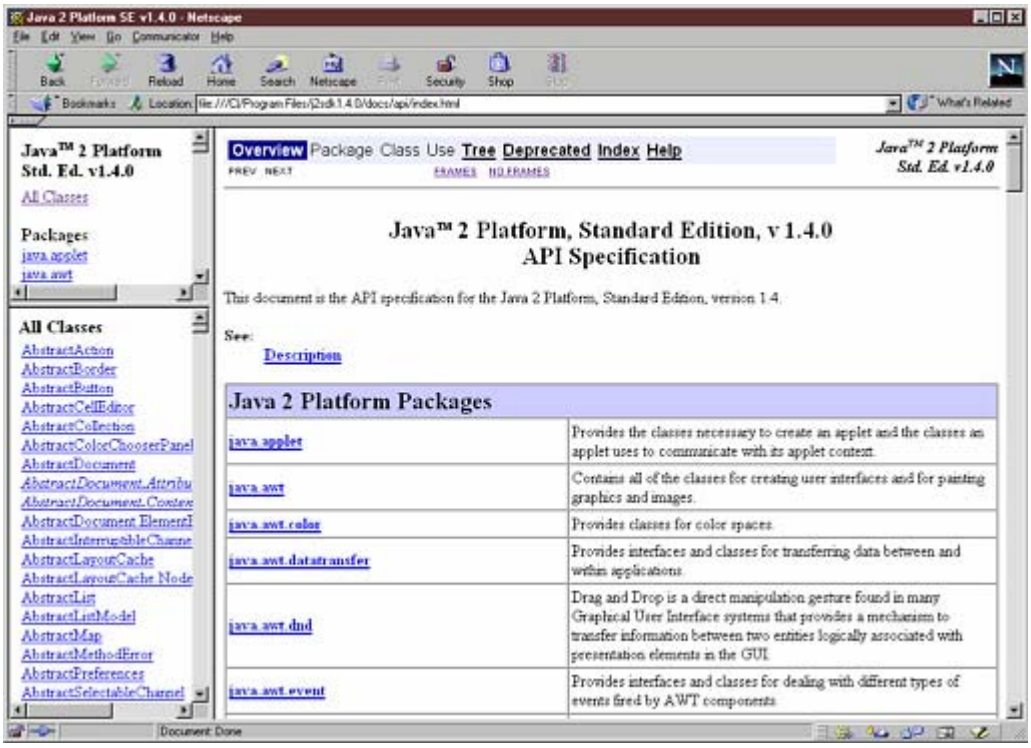
…i szczerze można powiedzieć, że nikt jej całej nigdy nie przeczyta - szczególnie, że jest po angielsku ☺.

(3.2) Java 2 Platform API Specification

Szczęśliwie, do naszych celów tak na prawdę istotne są tylko dwa punkty. Pierwszym jest:

API & Language Documentation
Java 2 Platform API Specification

Po kliknięciu w ten punkt naszym oczom ukaże się okno złożone z trzech ramek. W mniejszej ramce z lewej strony u góry znajduje się lista wszystkich pakietów, czyli bibliotek Javy. W większej ramce z lewej strony poniżej znajduje się lista wszystkich klas Javy. W dużej ramce po prawo umieszczona jest tabelka ze szczegółowym opisem wymienionych klas.



Kliknięcie w nazwę którejś biblioteki w małej ramce u góry z lewej powoduje załadowanie do ramki poniżej listy elementów zaimplementowanych w danym pakiecie (interfejsów, klas, wyjątków). Natomiast kliknięcie w nazwę konkretnej klasy czy wyjątku powoduje załadowanie do dużej ramki dokładnego opisu tego elementu…

…np.: opis klasy **String** zawartej w pakiecie **java.lang**. W ten sposób można znaleźć opis dowolnego elementu zawartego w standardowych bibliotekach Javy.

(3.3) Demonstration Applets and Applications

Drugim ważnym punktem dokumentacji jest:

Demos, Tutorials, Training, and Reference
Demonstration Applets and Applications

zawierający przykładowe aplety i aplikacje. Warto zapoznać się z tym punktem. Niektóre przykłady są bardzo widowiskowe.

dokumentacja	zbiór plików zawierających informacje dotyczące danego języka programowania, jego struktury, podstawowych elementów i sposobu używania.
HTML	*Hypetext Markup Language*, ogólnie przyjęty format hierarchicznych i multimedialnych dokumentów publikowanych w Internecie.
IDE	*Integrated Development Environment*, zintegrowane środowisko programistyczne, zestaw aplikacji niezbędnych do tworzenia, kompilacji, uruchamiania i testowania programów komputerowych.
Java	język programowania obiektowego stworzony przez amerykańską firmę Sun Microsystems.
JDK	*Java Development Kit*, zestaw aplikacji niezbędnych do kompilacji i uruchamiania programów napisanych w Javie. JDK zawiera w sobie JRE.
JRE	*Java Runtime Environment*, środowisko uruchomieniowe Javy, zestaw aplikacji niezbędnych do uruchamiania (ale nie kompilacji) programów napisanych w Javie. Zawiera maszynę wirtualną wraz z niezbędnymi bibliotekami.
PDF	*Portable Document Format*, stworzony przez amerykańską firmę Adobe wygodny format dokumentów odtwarzanych przez darmową przeglądarkę.
programowanie obiektowe	nowy styl programowania będący przedmiotem niniejszego podręcznika.