Język Java i technologie Web

Wykład semestralny

Tematy wykładów

1. Wprowadzenie
2. Programowanie imperatywne w Javie
3. Programowanie obiektowe
4. Programowanie obiektowe cz. 2
5. Wyjątki i programowanie wielowątkowe
6. Programowanie w sieci
7. Biblioteki AWT i Swing
8. Strumienie i operacje we-wy
9. Serializacja, wzorce projektowe
10. Powtórzenie wiadomości na egzamin

Materiały obowiązkowe

dostępne online za darmo!!!

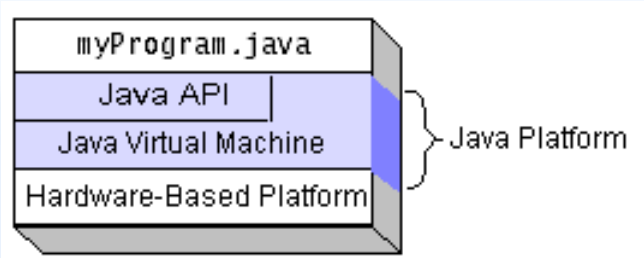
- Dokumentacja JDK 7 (w tym dokumentacja API) http://download.oracle.com/javase/7/docs/
- Oficjalny tutorial Oracle (Sun), http://download.oracle.com/javase/tutorial/
- Programowanie obiektowe – kurs EFS *Opracowanie programów nauczania na odległość na kierunku studiów wyższych – Informatyka* , http://wazniak.mimuw.edu.pl/index.php?title=Programowanie_obiektowe
- Bruce Eckel, Thinking in Java, Wydanie 4, Helion, 2011. http://mindview.net/Books/TIJ4

Materiały uzupełniające

- Cay Horstman, Gary Cornell, Java 2. Podstawy.
- Cay Horstman, Gary Cornell, Java 2. Techniki zaawansowane.
- Robert C. Martin, Czysty kod, Helion, 2010.
- Joshua Bloch, Java. Efektywne programowanie, Wydanie 2, Helion, 2009
- Andrzej Marciniak, JavaServer Faces i Eclipse Galileo. Tworzenie aplikacji WWW, Helion 2010.

Wprowadzenie

Pojęcie *technologii Javy* odnosi się zarówno języka programowania jak i platformy (sprzętowego lub programistycznego środowiska na którym programy są uruchamiane).



Java – rezult prac projektowych związanych z obsługą urządzeń pracujących w czasie rzeczywistym. Urządzenia te potrzebowały języka, który generuje niewielki kod wynikowy i jest niezależny od sprzętu. Pierwotnie, w systemach obsługujących te urządzenia używano języka C++, jednak ze względu na niedoskonałości kompilatorów C++ i różnice występujące między poszczególnymi jego realizacjami, powstał pomysł stworzenia nowego języka, całkowicie niezależnego od sprzętu i systemu operacyjnego.

Różnice między C++ a Javą 1.0

- narzucono ograniczenia, które ułatwiają testowanie programów i czynią kod przejrzystym,
- usunięto wskaźniki i uproszczono model zarządzania zasobami, np. pamięcią operacyjną,
- usunięto niektóre konstrukcje języka C++ (np.: **typedef**, #define, goto, **struct**, **union**, **enum**) oraz preprocesor, jako elementy niezgodne z paradygmatem programowania zorientowanego obiektowo,
- usunięto możliwość przeciążania operatorów,
- usunięto automatyczną konwersję zmiennych w przypadkach gdy wiąże się to z utratą precyzji,
- usunięto wielodziedziczenie (zastępując je mechanizmem implementacji interfejsów),
- zwolniono programistę od konieczności alokacji i zwalniania pamięci,
- zabroniono definiowania procedur i funkcji nie związanych z definicją żadnej klasy,
- wynikiem kompilacji programów Javy jest kod pośredni nazywany **kodem bajtowym** Javy (ang. Java Byte Code), a nie kod maszynowy, jak w C++

Właściwości JAVY - prostota

Prosty (ang. *simple*) – łatwy do nauczenia w bardzo krótkim czasie, zwłaszcza dla programistów znających C++. Oprócz usunięcia wielu kłopotliwych dla początkujących programistów cech języka C++ (jak omówione na poprzednim slajdzie), wpowadzono mechanizm automatycznego zarządzania pamięcią (ang. *garbage collection*) polegający na okresowym zwalnianiu nieużywanych zasobów pamięci.
Z prostoty składni języka wynika również ograniczenie miejsca potrzebnego do samodzielnego uruchamiania programów napisanych w Java. Ma to ogromne znaczenie przy uruchamianiu niezależnych aplikacji na małych komputerach oraz w Internecie.

Właściwości Javy – obiektowość

Zorientowany obiektowo (ang. *object-oriented*) – projektowanie obiektowe jest techniką, która kładzie nacisk na dane (obiekty) oraz interfejsy do nich. Klasa stanowi narzędzie do tworzenia nowych typów danych, których elementy noszą nazwę obiektów. Definicja klasy jest jedynym sposobem zdefiniowania nowego typu danych w Javie.

Otwarty na sieć (ang. *network-savvy*) – posiada bogate biblioteki procedur dla protokołów TCP/IP (FTP, HTTP). Aplikacje Javy mogą używać zasobów sieciowych poprzez adres URL z taką samą łatwością jak w przypadku lokalnych systemów plików.

Właściwości Javy - odporność

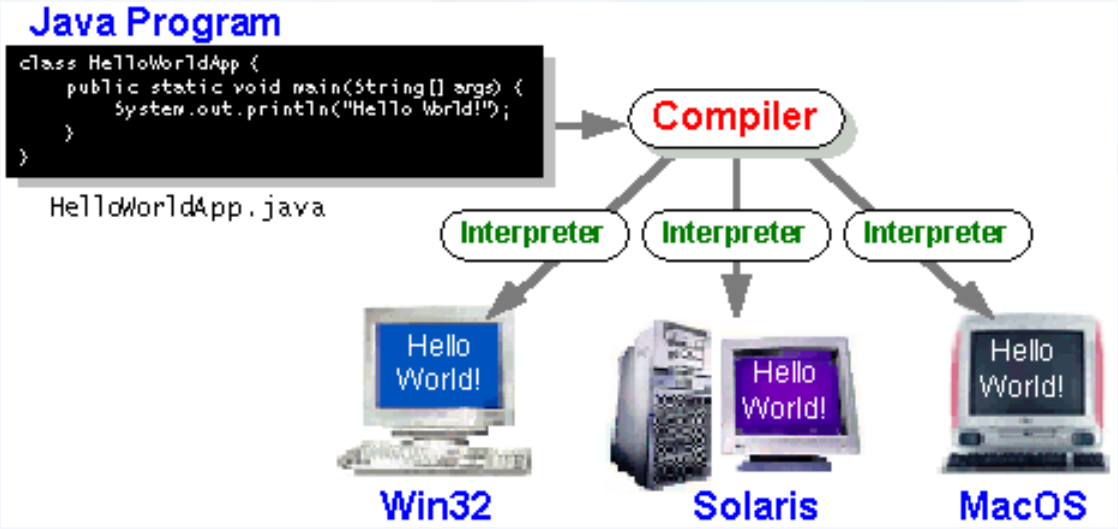
Odporny (ang. *robust*) – poświęcony jest duży nacisk na wczesne sprawdzanie możliwych błędów (podobnie jak w C++), sprawdzanie dynamiczne w trakcie pracy oraz eliminowanie sytuacji które mogą skłaniać do wystąpienia błędu. Niestety w C++ występuje wiele dziur w trakcie sprawdzania kodu podczas kompilacji (szczególnie podczas deklaracji metod), które w Javie nie występują gdyż Java wymaga jawnych deklaracji. Ponadto usunięcie jawnych wskaźników (lub też model wskaźnika w Javie) powoduje że eliminuje się możliwość nadpisywania pamięci i przekłamywania danych. Zamiast arytmetyki wskaźników, Java wykorzystuje prawdziwe tablice co pozwala na sprawdzanie indeksów. Programy w Javie nie mogą również wykorzystywać nieautoryzowanego dostępu do pamięci, co zdarza się w C/C++.

Właściwości Javy - bezpieczeństwo

Bezpieczny (ang. *secure*) – Java zostala zaprojektowana jako język dla systemów sieciowych/rozproszonych, co spowodwalo duży nacisk na problem bezpieczeństwa – oprogramowanie ma być wolne od wirusów i możliwości majstrowania przez niepowołane osoby. Techniki autoryzacji są oparte na kluczu publicznym. Zmiany semantyki wskaźników powodują, iż niemożliwy jest dostęp z innych aplikacji do struktur danych lub prywatnych danych w obiektach do których aplikacje te nie powinny mieć dostępu, co zamyka drogę dla największej formy działalności wirusów (*The Java Language. An overview*, Gosling i McGilton, 1996).

Właściwości Javy - otwartość

Neutralny (ang. *architecture neutral*) – kompilator generuje instrukcje w kodzie bajtowym, który jest wykonywalny na wielu maszynach pod warunkiem obecności maszyny wirtualnej Javy. Instrukcje w kodzie bajtowym nie mają nic wspólnego ze szczególną architekturą komputera. Są raczej zaprojektowane tak, by być łatwo interpretowane na dowolnej maszynie i szybko tłumaczone na kod maszynowy danego sprzętu.

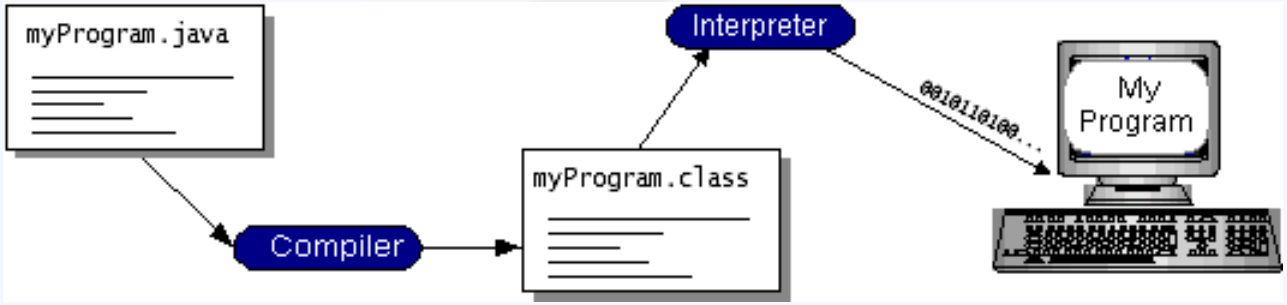


Właściwości Javy - przenośność

Przenośny (ang. *portable*) – już sam fakt wykorzystania otwartej architektury determinuje przenośność kodu. Ponadto w przeciwieństwie do C/C++ nie występują problemy z zależnością implementacyjną (od środowiska). Przykładowo, rozmiary podstawowych typów danych są zawsze takie same, czyli 'int' to zawsze 32 bitowa zmienna całkowita, a 'float' to 32 bitowa zmienna rzeczywista wg standardu IEEE 754. Zasadniczo wszystkie liczące się CPU dzielą te ch-ki. Również biblioteki, będące częścią systemu definiują przenośne interfejsy. Przykładowo, istnieje abstrakcyjna klasa Windows oraz jej implementacje na Unix, Macintosh czy Windows NT.

Właściwości Javy - intrepretowalność

Interpretowany (ang. *interpreted*) – kod bajtowy Javy (Java bytecode) jest tłumaczony na bieżąco na kod maszynowy danego komputera (interpretowany). Kompilacja kodu występuje tylko raz, natomiast interpretacja zawsze gdy tylko program jest uruchamiany. Kod bajtowy Javy nie jest wykonywany tak szybko, jak zwykły, napisany w kodzie maszynowym procesora program. Aby rozwiązać ten problem wprowadzono "kompilację w locie" (ang. Just-in-Time compilation JIT) kodu bajtowego do kodu maszynowego danego procesora.



Właściwości Javy - wielowątkowość

Wielowątkowy (ang. *multithreaded*) –Jeśli program napisany wielowątkowo wykonywany jest na maszynie wieloprocesorowej, to różne wątki mogą być wykonywane w tym samym czasie na różnych procesorach. Sterowanie programu w takich przypadkach przebiega współbieżnie (ang. *concurrent*). Na komputerach jednoprocesorowych wykonanie programów wielowątkowych jest tylko emulowane. Emulacja ta polega na naprzemiennym przydzielaniu czasu procesora poszczególnym wątkom wg. pewnego algorytmu synchronizacji. Java posiada bogaty zestaw procedur synchronizacji wątków, pochodzących z systemu Xerox MESA/CEDAR.

Właściwości Javy - dynamiczność

Dynamiczny (ang. *dynamic*) – język Java jest bardziej dynamicznym językiem od C/C++, jako że został zaprojektowany tak, aby adaptować się do rozwijających się środowisk.
Przykładowo, wielkim problemem w C/C++ jest problem implementacji kodu. Jeśli firma A sprzedaje bibliotekę klas (np. komponentów plug&play), a firma B korzysta z tych klas w swoich produktach, to jeśli A zmieni swoją bibliotekę i wprowadzi nowy produkt, wówczas B musi przekompilować i przedystrybuować swój software. Jeśli końcowy uzytkownik używa zarówno oprogramowania obu firm niezależnie, to pojawia się problem (np. A dostarcza OS, a B dostarcza aplikacje).
Java unika tego problemu poprzez uproszczenie paradygamtu programowania obiektowego. Do bibliotek wolno dopisywać nowe metody bez efektu na klientach.

Problem dynamiki języka jest oczywiście dużo bardziej złożony, a rozwiązania zastosowane w Javie są omówione dalej.

Właściwości Javy - wydajność

Wydajny (ang. *High performance*) – Proces właściwego generowania kodu maszynowego z kodu bajtowego jest bardzo prosty, gdyż format kodu bajtowego został zaprojektowany z myślą o optymalizacji kodu maszynowego. W rezultacie już w trakcie kompilacji następuje optymalizacja kodu.

Przykładowo, w kodzie interpretowanym przez Sun Microsystems SPARCStation 10, w ciągu sekundy wywołuje się 300 000 metod. Pod względem szybkości kod bajtowy konwertowany na maszynowy jest niemal tak szybki jak zwykły kod maszynowy C/C++.

Właściwości Javy - wydajność

Słabą stroną programów napisanych w Javie była mała szybkość wykonywania spowodowana interpretowaniem kodu bajtowego w przeglądarce. Aby rozwiązać ten problem wprowadzono "kompilację w locie" (ang. Just-in-Time compilation JIT) kodu bajtowego do kodu maszynowego danego procesora. Kiedy przeglądarka ładuje pierwszy raz aplet Javy, i jeśli opcja JIT jest aktywna, to przekształca ona kod bajtowy w kod maszynowy danego procesora, który zapisuje do pliku dyskowego. Przeglądarka wykonuje wtedy kod maszynowy. W efekcie, kompilator JIT traktuje kod bajtowy jako język źródłowy, który kompiluje do postaci programu w języku maszynowym lokalnej maszyny. Ponieważ w kodzie bajtowym nie było w trakcie kompilacji do kodu maszynowego danego procesora żadnych zmian, więc w rezultacie program ciągle jest bezpieczny i niezależny od sprzętu. Kompilacja JIT powoduje, że program napisany w Javie pracuje kilka razy szybciej niż wykonywany przez najlepszy interpreter kodu bajtowego.

Ocena Javy na tle innych języków

	Java	SmalTalk	ICL	Perl	C	C++
prosty (simple)	■	■	■	▒	▒	□
zorientowany obiektowo (object oriented)	■	■	□	■	□	▒
solidny (robust)	■	■	■	■	□	□
bezpieczny (secure)	■	▒	▒	■	□	□
interpretowany (interpreted)	■	■	■	■	□	□
dynamiczny (dynamic)	■	■	■	■	□	□
przenośny (portable)	■	▒	■	■	▒	▒
neutralny (neutral)	■	▒	▒	■	□	□
wątki (threads)	■	□	□	■	□	□
automatyczne zwalnianie pamięci (garbage collection)	■	■	□	□	□	□
wyjątki (exceptions)	■	■	□	■	□	▒
wydajność (performance)	wysoka	średnia	niska	średnia	wysoka	wysoka

Realizacja atrybutu: pełna ■
częściowa ▒
brak □

Co można napisać w Javie?

Aplikacje – samodzielne programy uruchamiane na platformie Javy (w tym serwery, tj. aplikacje służące klientom w sieci np. Web serwery, proxy, serwery pocztowe, serwery drukarek itp.)

Applety – programy „doklejane” do stron internetowych i odczytywane za pomocą przeglądarek z zainstalowanymi interpreterami Javy,

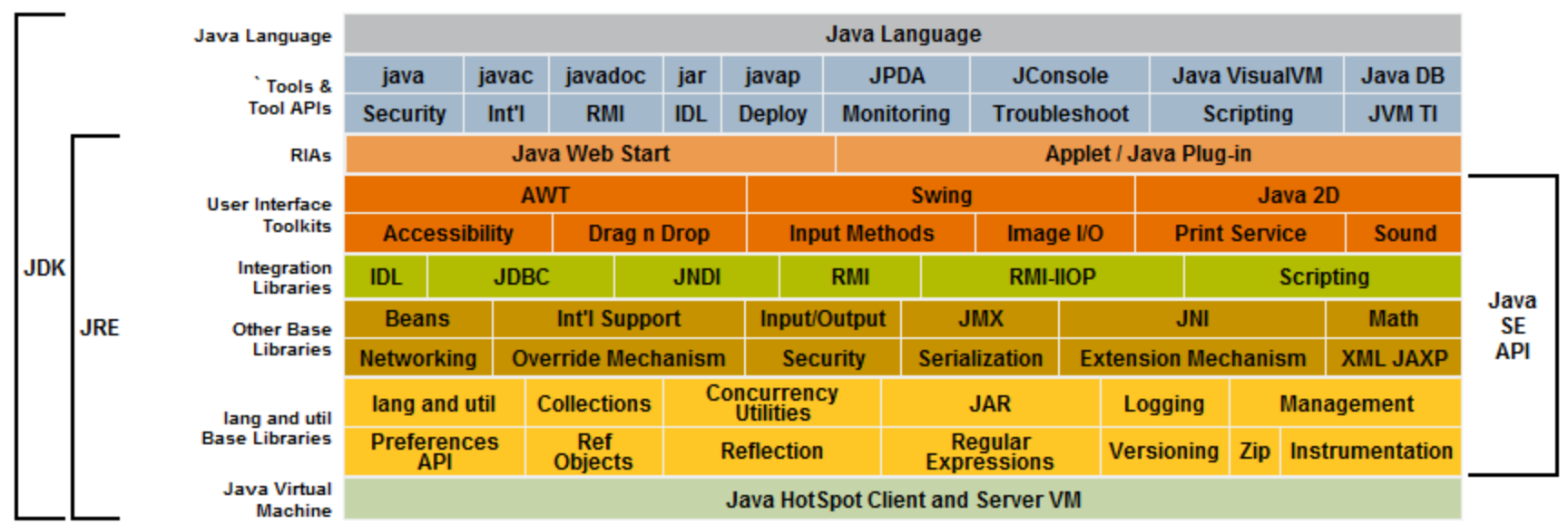
Servlety – programy które podobnie jak applety są stworzone dla potrzeb Internetu, ale uruchamiane są po stronie serwera. Mogą być wykorzystane do budowy interaktywnych aplikacji internetowych, zastępując skrypty CGI.

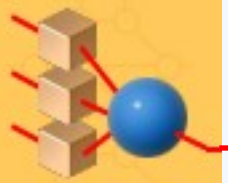
Midlety – programy uruchamiane w urządzeniach PDA, tel. itp.

Elementy pełnej platformy Java JDK

- **zasadnicze**: obiekty, łańcuchy, wątki, liczby, obsługa wej.-wyj., struktury danych, obsługa właściwości systemu oraz czasu i daty, etc.
- **applety**: zbiór konwencji używanych przez applety.
- **usługi i protokoły sieciowe**: URLs, TCP (Transmission Control Protocol), UDP (User Datagram Protocol) sockets, adresy IP (Internet Protocol).
- **lokalizacja**: programy mogą automatycznie adaptować się do regionu geograf. i być wyświetlane w odpowiednim języku
- **bezpieczeństwo**: zarówno niskiego i wysokiego poziomu, włączając elektroniczne sygnatury, zarządzanie kluczem publ. I pryw., kontrola dostępu i certyfikaty.
- **komponenty software'owe**: określane jako JavaBeans™, mogą podłączać się do istniejących komponentów.
- **serializacja obiektów**: pozwala na komunikację Remote Method Invocation (RMI).
- **Java Database Connectivity (JDBC™)**: dostarcza dostęp do szerokiego spektrum baz relacyjnych.

Java Platform SE 7





Bazowe systemy i biblioteki Javy

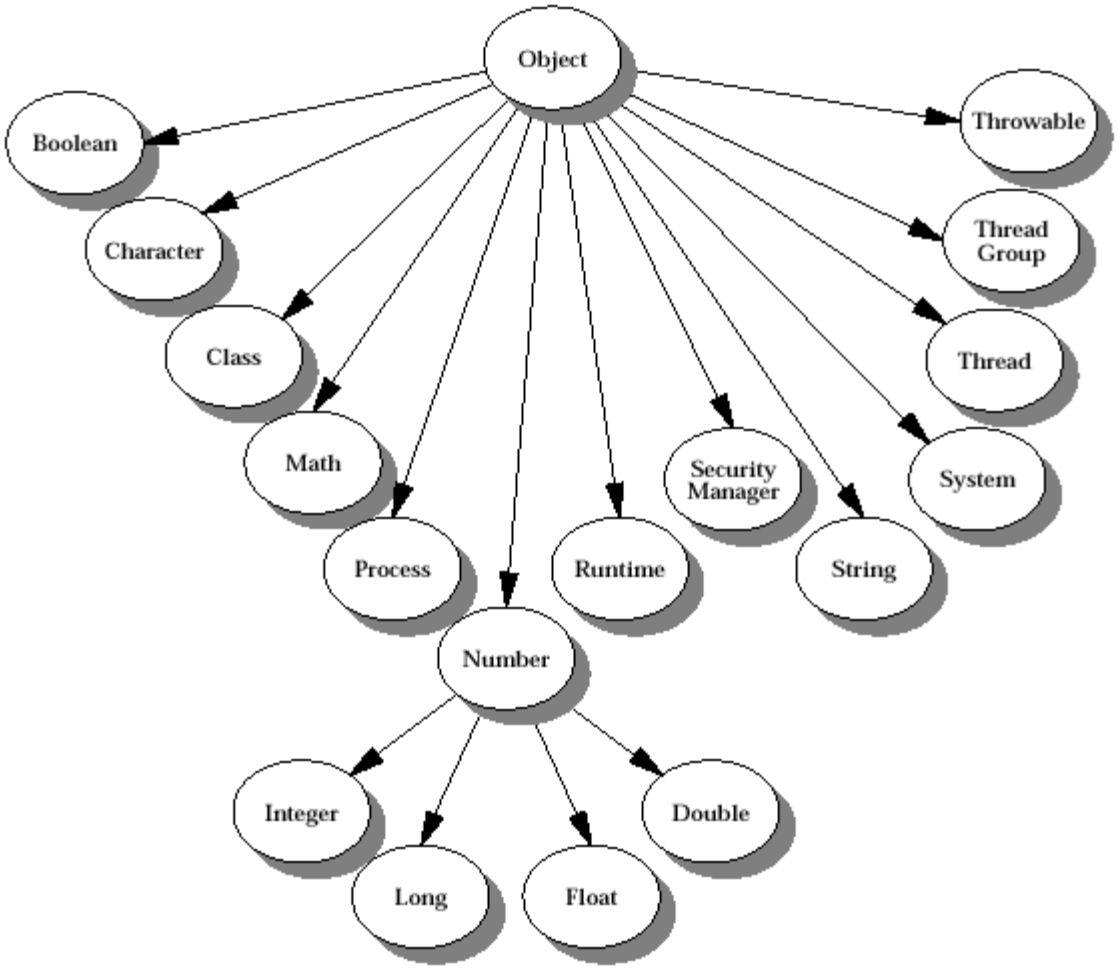
Basic Java language classes	—	java.lang
The Input/Output package	—	java.io
The Java Utilities package	—	java.util
The Abstract Window Toolkit	—	java.awt

Java.lang

Pakiet zawiera zbiór typów bazowych (typów języka) które są zawsze importowane do dowolnego kompilowanego kodu. Tam można znaleźć deklarację (korzeń hierarchii klas) i , oraz wątków, wyjątków, podstawowych typów danych oraz fundamentalnych klas.

Uwaga: Klasy takie jak np. , , i są „opakowaniami” (ang. “wrapper”) dla podstawowych typów.

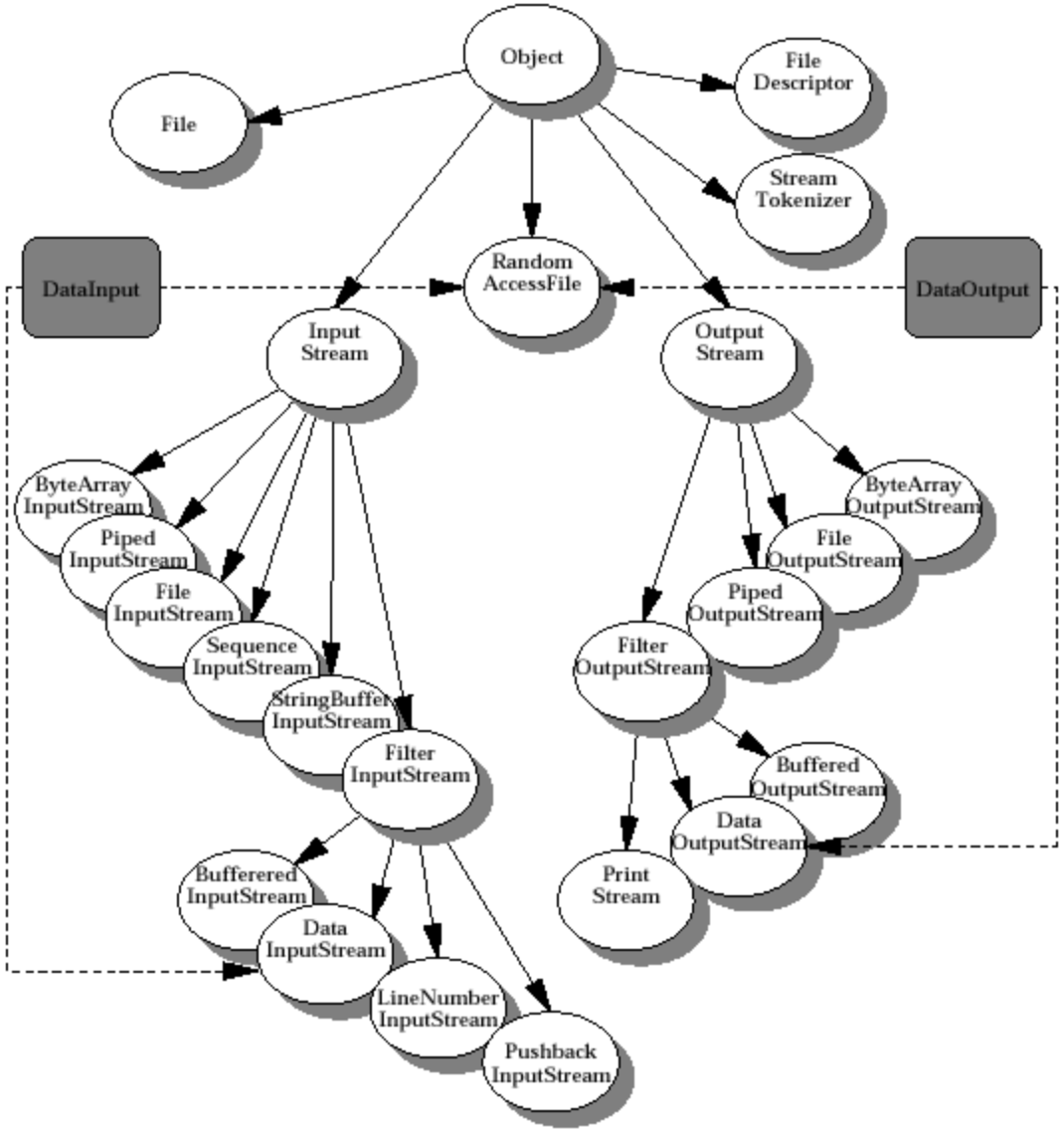
Java.lang



Klasy java.lang, wyłączając wyjątki i klasy błędów, tj. Throwable i Thread są korzeniami całej hierarchii.

Java.io

Pakiet zawiera deklaracje klas zarządzających strumieniami i dostępem do plików. Tu znajdują się odpowiedniki Standard I/O Library systemów UNIX'owych. Dalsza biblioteka , dostarcza narzędzia dla socket'ów, telnet interfaces, i URLs.



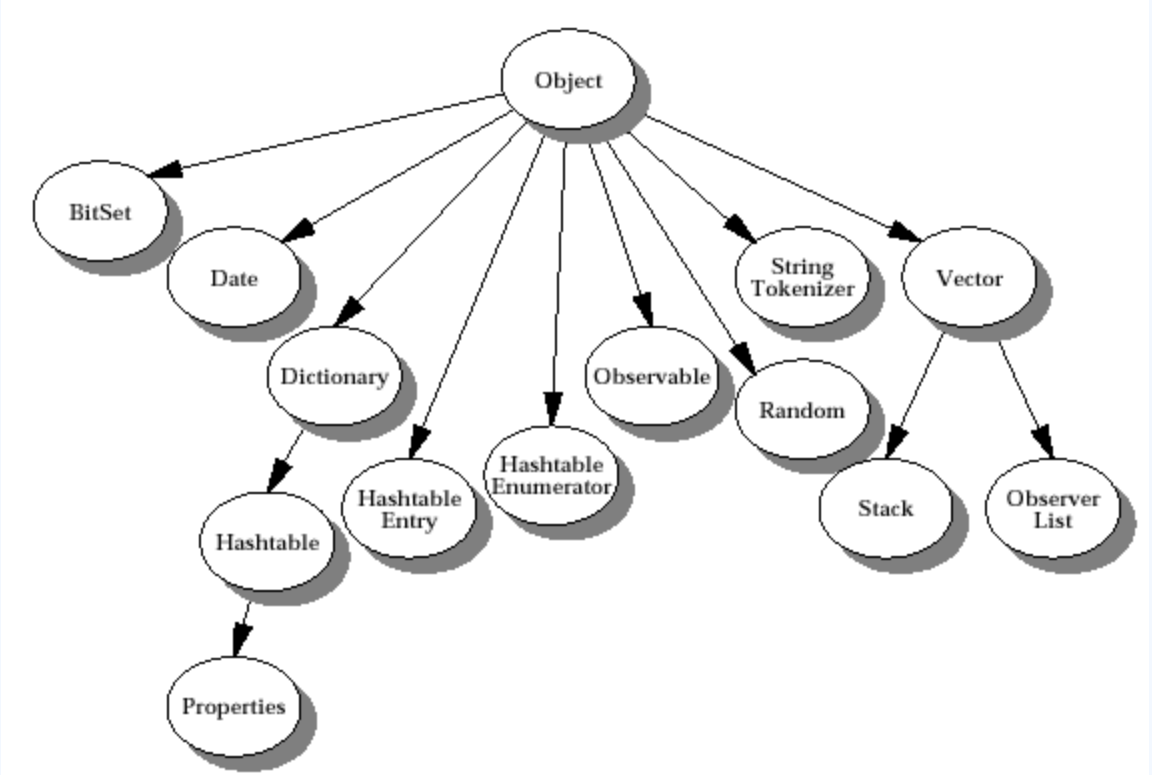
Java.io

Szare elementy oznaczają interfejsy, a nie klasy.

O interfejsach dowiesz się na następnych wykładach.

Java.util

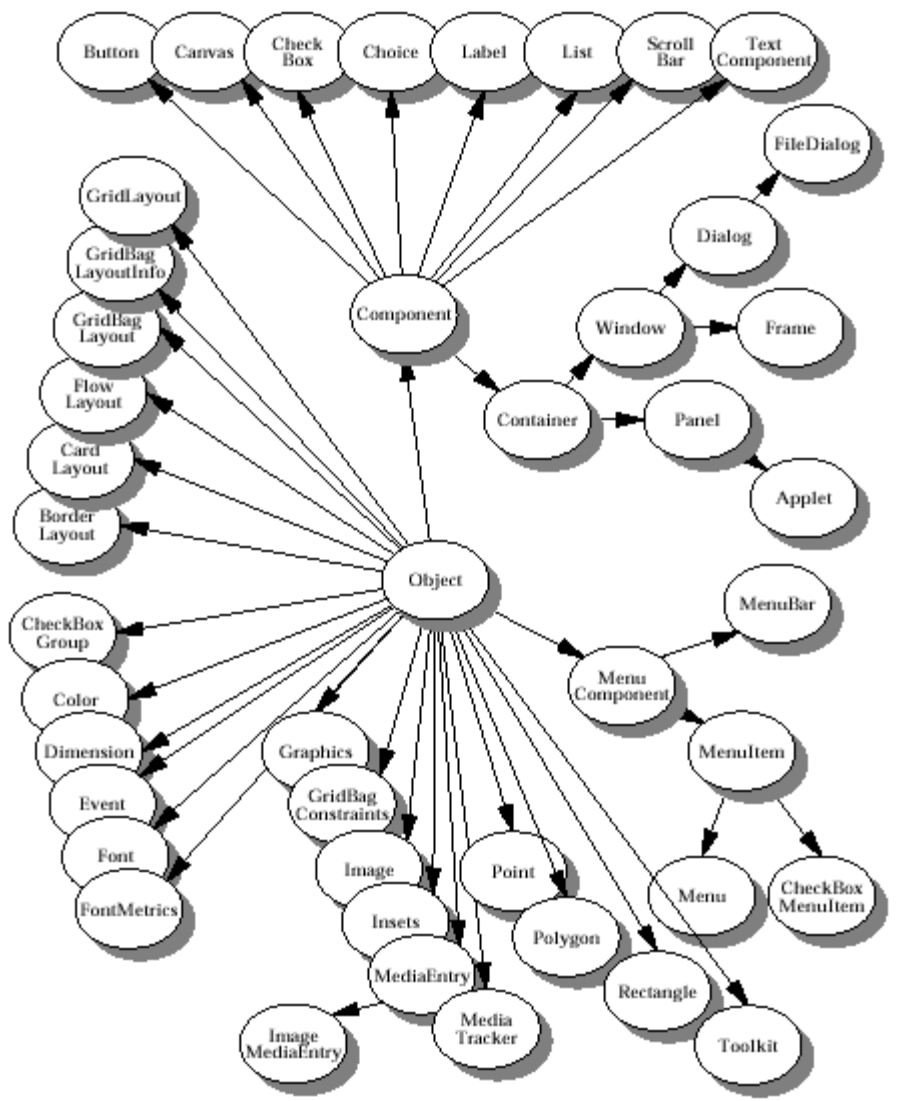
Pakiet zawiera różne użyteczne klasy włączając zbiór klas takich jak Dictionary i Vector.



Java.awt

Abstract Windowing Toolkit dostarcza komponenty graficznego interfejsu, takie jak przyciski, zdarzeina, kolory, fonty itp.

Hierarchia Java.awt



Pierwsza aplikacja w Javie

- Utwórz plik źródłowy z rozszerzeniem .java. Możesz wykorzystać dowolny edytor tekstowy.

- Skompiluj źródło do kodu bajtowego z rozszerzeniem .class. Kopilator javac generuje kod bajtowy tak aby był zrozumiany w maszynie wirtualnej.

- Uruchom program na maszynie wirtualnej.

Pierwsza aplikacja – „witaj” (Sun)

```
/** * Klasa Witaj implementuje program wyświetlający na ekranie
napis Witaj do standardowego wyjścia */

public class Witaj
{ public static void main(String[] args)
{  System.out.println(„Witaj"); }
}
```

Zapisz powyższy tekst do pliku z rozszerzeniem .java o nazwie identycznej jak nazwa klasy zawierającej funkcję main - program.
Następnie w wierszu poleceń systemu skompiluj poleceniem

```
javac Witaj.java
```

po czym uruchom program poleceniem

```
java Witaj
```

Podstawowe elementy programu

Komentarze: /* mogą obejmować wiele linii kodu*/
// lub tylko do końca linii ,
/** a mogą nawet służyć do automatycznego generowania dokumentacji.
(w Java Sun jest to moduł javadoc)*/

class Witaj { ...} - definicja klasy

public static void main(String[] args){...} – każda aplikacja musi zawierać funkcję main, a klasa ją zawierająca powinna nazywać się identycznie jak nazwa pliku.

- Public informuje funkcja może być wywołana z zewnątrz klasy,
- static informuje że funkcja jest metodą klasy a nie metodą instancji (obiektu).
- void oznacza że nie zwraca wartości.
- String [] args oznaczają listę parametrów dla wywołania wsadowego z linii poleceń

Niezbędne informacje.

System.out.println(„Witaj"); - wyświetla napis – metoda ***println*** należy do obiektu ***out*** będącego egzemplarzem klasy ***OutputStream*** i inicjalizowanym statycznie w klasie ***System***. Zanim zostaną omówione informacje dotyczące powyższych klas, metodę *System.out.println* należy traktować jako odpowiednik konstrukcji ***cout<<*** z języka C++.

Uwaga na duże i małe litery. Odstępy w Javie nie grają roli, tzn. cały omawiany program mógłby być napisany jak poniżej:

```
public
class
Witaj
{
public
.....
```

Pierwszy applet

```
import java.applet.*;
import java.awt.*;

/**
 * Klasa WitajApplet implementuje wyświetlanie
 * "Witaj!".
 */
public class WitajApplet extends Applet {
    public void paint(Graphics g) {
        // Wyświetlanie "Witaj!"
        g.drawString("Witaj!", 50, 25);
    }
}
```

Zapisz plik jako WitajApplet.java. Następnie skompiluj analogicznie do poprzedniego przykładu (otrzymasz WitajApplet.class)

Osadź applet w dokumencie HTML.

```
<HTML>
<HEAD>
<TITLE>Pierwszy aplecik</TITLE>
</HEAD>
<BODY>
Oto  mój pierwszy program:
<APPLET CODE="WitajApplet.class"
WIDTH=150 HEIGHT=25>
</APPLET>
</BODY>
</HTML>
```

Podgląd appletu

Gotową stronę HTML z działającym appletem można przeglądać na dwa sposoby: za pomoca przeglądarki WWW lub przy pomocy przeglądarki appletów. W pakiecie Java 2 firmy Sun Microsystems, służy do tego programik appletviewer, czyli

appletviewer strona.html

Budowa Appletu WitajApplet

```
import java.applet.*; // applet importuje
niezbędne
import java.awt.*;    // klasy i pakiety


public class WitajApplet extends Applet {
```

Każdy applet musi mieć zdefniowaną podklasę klasy Applet, w tym przypadku klasa WitajApplet dziedziczy wiele metod klasy Applet

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

Applet nie musi mieć funkcji main, ale powinien mieć zaimplementowaną jedną z trzech funkcji: ***init***, ***start*** lub ***paint***.

Pytania kontrolne

1. Wymień najistotniejsze róznice między językiem C++ a Javą.
2. Wymień najistotniejsze właściwosci języka Java.
3. Jakiego typu programy można napisać w Javie.
4. Wymień cztery podstawowe pakiety klas w Javie.
5. Scharakteryzuj strukturę aplikacji w Javie.
6. Scharakteryzuj strukturę appletu.