Spis treści

1 Java^{TM} **1**

2 Co to jest Platforma Java^{TM} **1**

3 Przygotowanie komputera **2**

4 Pierwszy program **2**

5 Dokumentacja **3**

6 Budowa aplikacji. Klasy. **3**

7 Pola i metody **4**

8 Konstruktory **5**

9 Inne proste przykłady aplikacji **6**

Temat: Uruchamianie programów w Java^{TM}
Celem pierwszego wykładu jest napisanie prostego programu w Java^{TM}. Jego zadaniem będzie wyprowadzenie na ekran zadanego ciagu znaków - napisu.

Podstawa: Ken Arnold, James Gosling, Java^{TM}, WNT, Warszawa 1999; Monica Pawlan, Basic Java 1 & Basic Java 2

1 Java^{TM}

Twórcą języka Java^{TM} jest James Gosling, obecny wiceprezes firmy Sun Microsystems. Java^{TM} posiada najlepsze cechy współczesnych języków programowania, Jako uniwersalny język programowania posiada wiele możliwości pozwalających budować duże programy bądź wykorzystujące sieć bądź od niej niezależne. Łatwość programowania w Java^{TM} i możliwość wykonywania programów na odległych komputerach w sposób bezpieczny są niezwykle istotne w zastosowaniach Java^{TM} związanych z siecią.

Java^{TM} posiada rozwinięte mechanizmy programowania obiektowego (klasy, obiekty, dziedziczenie, możliwość rozszerzania klas itp), wbudowana obsługę wyjątków (błędów), możliwość tworzenia programów wielowątkowych, bogate srodowisko graficzne oraz możliwość obsługi zdarzeń (okna, klawiatura, mysz).

2 Co to jest Platforma Java^{TM}

Platforma Java^{TM} to API - application programing interface oraz JVM -Java^{TM} virtual machine.

API to zbiór gotowych podprogramów w Java^{TM}, które można wykorzystać w konkretnych zadaniach. Pozwala to zaoszczędzic czas przy tworzeniu programów.

Programy napisane w Java^{TM} uruchamiane sa (interpretowane) za pomocą JVM - wirtualnej maszyny Java^{TM}. Można je uruchamiać na dowolnych komputerach wyposażonych w JVM. Jest to wielka zaleta tego rodzaju programów (podobnie `perl`).

Na przykład, program napisany na komputerze Sun z systemem operacyjnym Solaris może być uruchomiony na komputerze PC z systemem `WINDOWS` lub na komputerze z systemem `LINUX` lub odwrotnie.

3 Przygotowanie komputera

Zanim zaczniemy pisać nasze programy w Java^{TM} musimy przygotowac środowisko Java^{TM}. W tym celu musimy udac sie na stronę domowa firmy Sun gdzie powstała Java^{TM}. Tam znajdziemy odpowiednie oprogramowanie, a więc Java 2 SDK, wiele cennych podręczników w wersji elektronicznej, przykłady itp.

4 Pierwszy program

Program piszemy używając dowolnego (dobrego) edytora tekstu. Zapisujemy go pod nazwa `Przyklad.java`. Należy przy tym zwracać uwagę na wielkość liter w nazwie jak tez w samym programie. Zauważmy, że zarówno nazwa pliku zawierającego program jak i nazwa klasy głównej (jedynej w tym programie) są takie same, a mianowicie `Przyklad`. Tak to zostało ustalone i ma tak być.

```
// ---------------------------------------------------------------------
/*
Fizyka komputerowa, IV, 2001.  Java(TM).
Program #1.
A. Baran, IFiz UMCS, 2000.
http://tytan/umcs.lublin.pl/baran
*/
// ---------------------------------------------------------------------
//

//Prosty przykład
class Przyklad {
  public static void main(String[] args){
    System.out.println("Bardzo prosty program w Java.");
```

```
  }
}
```

Pierwsza linia programu, zaczynajaca sie od dwóch znaków `slash`, a wiec `//`, jest komentarzem. Następna linia jest deklaracją głównej funkcji programu. Powinna mieć nazwę `main`. Argumentem funkcji `main` jest standardowa tablica `args`, której elementami są łańcuchy (String), a więc ciągi znaków. Sa to te same łancuchy, ..., ale o tym dalej.

Linia `System.out.println("Bardzo prosty program w Java.");` powoduje wypisanie na ekran łańcucha `Bardzo prosty program w Java.` W linii tej skorzystaliśmy z funkcji `println` umieszczonej w module bibliotecznym `System.out`. Funkcja ta działa podobnie jak funkcja `writeln` języka `Pascal`.

Następny krok to kompilacja programu. W wierszu poleceń piszemy

```
javac Przyklad.java
```

Wynikiem jest plik `Przyklad.class`. Jest to tzw B-kod Java^{TM}. Nadaje sie on do interpretacji przez dowolną maszynę wirtualną Java^{TM}.

Wykonamy ten program poleceniem

```
java Przyklad
```

Proszę zwrócić uwagę na brak rozszerzenia nazwy pliku. Wynikiem jego działania jest linia tekstu: `Bardzo prosty program w Java.`

5 Dokumentacja

Oprócz środowiska programistycznego Java 2 SDK zawiera obszerną dokumentacją dotycząca API jak też podręcznik do nauki Java^{TM}.

> Temat: Budowa aplikacji
> Celem tej części zajęć jest omówienie budowy aplikacji w Java^{TM}.

6 Budowa aplikacji. Klasy.

Program w Java^{TM} zbudowany jest z klas (class). Klasa jest podobna w swojej budowie do klasy w Delphi lub do struktury (structure) w `C++` i jest trochę podobna do `record`u `Pascal`a. Różnica między rekordem, a klasą polega między innymi na tym, że klasa oprócz pól danych (jak w rekordzie) posiada metody. Metody są to funkcje i procedury operujące na danych zgromadzonych w klasie. Mogą więc one ustawiać pola danych, podawać je, coś obliczać itp.

W pierwszym naszym przykładzie klasa `Przyklad` nie zawiera żadnych pól danych natomiast zawiera jedna metodę o nazwie `main`. Deklaracja tej metody jest następująca

```
public static void main(String[] args) {
  ////...
}
```

Słowo `public` oznacza, że metoda jest dostępna spoza modułu w którym jest deklarowana. Słowo `void` znaczy tyle co nic, tzn., że metoda `main` nie zwraca do wołającego ją podprogramu żadnej wartości. Jeśli by tak było należałoby podać zamiast słowa `void` (określającego typ) typ zwracanej wielkości, a więc np. `int, double, string` itd.

Przybliżymy teraz sens deklaracji `static`. Jeśli słowo to wystąpi to JVM uruchamia metodę `main` bez tworzenia kopii, instancji klasy wzorcowej. Instancja jest wykonywalną kopia klasy. Można utworzyć wiele egzemplarzy tej samej klasy wzorcowej. Dokładniej zostanie to omówione dalej. Słowo `public` pozwala JVM uruchomic metodę `main` w taki właśnie sposób a więc bez kopiowania.

Metoda `main` wywołuje następnie statyczną metodę `println`, która znajduje się w klasie `System`. Klasa `java.lang.System` umie obsłużyc terminal i między innymi potrafi drukować. Jest ona zbudowana ze statycznych pól danych i metod.

W przypadku klas niestatycznych, przed uruchomieniem ich metod, należy utworzyć instancje klas, a następnie uruchomić metody na rzecz tychże.

7 Pola i metody

Czym różni się niżej wypisany program od poprzedniego?

```
// ---------------------------------------------------------------------
/*
Fizyka komputerowa, IV, 2001.  Java(TM).
Program #2.
A. Baran, IFiz UMCS, 2000.
http://tytan/umcs.lublin.pl/baran
*/
// ---------------------------------------------------------------------
//

////Prosty przykład 2
class Przyklad2 {
  static String tekst = "Bardzo prosty program w Java.";
  public static void main(String[] args){
    System.out.println(tekst);
  }
}
```

Oprócz metody `main` zadeklarowany został statyczny łańcuch (`static String`) o nazwie `tekst`. Jego wartością jest napis `"Bardzo prosty program w Java.`

. Napis ten zmienna `tekst` przekazuje do metody `println`. Wynik działania programu jest taki sam jak wcześniej. Pole zawierające łańcuch jest polem danych klasy `Przyklad2`.

W celach dydaktycznych skomplikujemy omawiany przykład jeszcze bardziej.

```
// ---------------------------------------------------------------------
/*
Fizyka komputerowa, IV, 2001.  Java^TM.
Program #3.
A. Baran, IFiz UMCS, 2000.
http://tytan/umcs.lublin.pl/baran
*/
// ---------------------------------------------------------------------
//

////Prosty przykład 3
class Przyklad3 {
  String tekst = "Bardzo prosty program w Java.";
  static String tekst2 = "Tekst statyczny";

  String pobierzTekst(){
    return tekst;
  }

  String pobierzTekstStat(){
    return tekst2;
  }

  public static void main(String[] args){
    Przyklad3 instancjaPrzyklad = new Przyklad3();
    String pobranyTekst = instancjaPrzyklad.pobierzTekst();
    String pobranyTekstStat = instancjaPrzyklad.pobierzTekstStat();
    System.out.println(pobranyTekst);
    System.out.println(pobranyTekstStat);
  }
}
```

Metody i zmienne statyczne nosza nazwę klasowych. Metody klasowe operują tylko na polach danych klasy do której należą. Metody, które są instancjami metod operują na polach klasowych oraz na polach instancji. Istnieje tylko jedna kopia danych klasowych, natomiast każdy egzemplarz klasy (instancja) posiada swoje pola danych. Jest to nieco zagmatwane.

8 Konstruktory

Klasy posiadaja specjalne metody zwane konstruktorami. Są one wywoływane gdy kreowana jest instancja klasy. Konstruktor posiada taką samą nazwę jak klasa i nie zwraca żadnej wartości. Konstruktory rezerwują zasoby dla tworzonego egzemplarza klasy. Oprócz tego mogą wykonywać różne prace wstępne, np. ustawiać dane, coś obliczać, itd. Jeśli konstruktor nie jest zadeklarowany wówczas Java^{TM} dostarcza bezargumentowego konstruktora o nazwie takiej jak nazwa klasy.

```
// ---------------------------------------------------------------------
/*
Fizyka komputerowa, IV, 2001.  Java(TM).
Program #4.
A. Baran, IFiz UMCS, 2000.
http://tytan/umcs.lublin.pl/baran
*/
// ---------------------------------------------------------------------
//

////Prosty przykład 4
class Przyklad4 {
  String tekst;

  // Konstruktor klasy
  Przyklad4() {
    tekst = "Bardzo prosty program w Java.";
  }

  // Metoda pobierania
  String pobierzTekst(){
    return tekst;
  }

  public static void main(String[] args){
    Przyklad4 instancjaPrzyklad = new Przyklad4();
    String pobranyTekst = instancjaPrzyklad.pobierzTekst();
    System.out.println(pobranyTekst);
  }
}
```

W klasie może istnieć więcej konstruktorów (o tej samej nazwie) różniących się liczbą lub typem argumentów.

9 Inne proste przykłady aplikacji

Program w następnym przykładzie oblicza kolejne wyrazy ciągu Fibonacciego. Dzieje się to w pętli `while(...){...}`. Działa ona tak jak np. pętla `while` w języku `Pascal`.

```
// ---------------------------------------------------------------------
/*
Fizyka komputerowa, IV, 2001.  Java(TM).
Program #5.
A. Baran, IFiz UMCS, 2000.
http://tytan/umcs.lublin.pl/baran
*/
// ---------------------------------------------------------------------
//

////  K. Arnold, J. Gosling
class Fibonacci {
   /** Ciąg Fibonacciego */
   public static void main(String[] args) {
     int lo = 1;
     int hi = 1;

     System.out.println(lo);
     while (hi < 50) {
       System.out.println(hi);
       hi = lo + hi;  // nowe lo
       lo = hi - lo;  /* nowe lo wynosi (suma - stare lo),
                         tzn. stare hi */
     }
   }
 }
```

Nieco lepszą wersją powyższego programu jest program pokazany niżej. Wprowadzono tu zmienną `MAX_INDEX`, która ogranicza zakres pętli `while`. Aby to miało miejsce, dodatkowa zmienna kontrolna `i` zlicza ilość składników ciągu Fibonacciego, zwiększając się za każdym razem wewnątrz pętli `while` w poleceniu `i++;`. Wykorzystano tutaj operator inkrementacji `++`, typowy dla języka `C++`.

```
// ---------------------------------------------------------------------
/*
Fizyka komputerowa, IV, 2001.  Java(TM).
Program #6.
A. Baran, IFiz UMCS, 2000.
http://tytan/umcs.lublin.pl/baran
*/
```

```
// ------------------------------------------------------------------
//

////  wg. K. Arnold, J. Gosling
class Fibonacci1 {
  /** Ciąg Fibonacciego */
  public static void main(String[] args) {
    long lo = 1;
    long hi = 1;
    int MAX_INDEX = 50;
    int i = 0;

    System.out.println(lo);
    while (i < MAX_INDEX) {
      System.out.println(hi);
      hi = lo + hi;  // nowe lo
      lo = hi - lo;  /* nowe lo wynosi (suma - stare lo),
                        tzn. stare hi */
      i++;
    }
  }
}
```