WPROWADZENIE DO JĘZYKA JAVA

podstawowe informacje: zarys historii, zasadnicze cechy i pojęcia Javy, wirtualna maszyna Javy (JVM), narzędzia Javy - zawartość pakietu Software Development Kit (SDK), pierwsze kroki w Javie - kompilacja i uruchomienie aplikacji.

FUW

Zarys historii

- ”Narodziny” Javy – 1990 rok, Sun Microsystems, Inc. Mountain View, California.
  - Początkowa nazwa nowego języka – *OAK* .
  - Przeznaczenie - programowanie inteligentnych urządzeń domowego użytku.
- Główny architekt Javy – James Gosling.
- W 1994 r. wstrzymano prace nad Javą. Przyczyna - brak zainteresowania projektem Green ze strony znanych producentów urządzeń elektronicznych.
- Rozwój Internetu powoduje ponowne zainteresowanie się językiem tworzącym przenośne programy.
  - 1995 rok, konferencja w San Francisco - pierwsza publiczna prezentacja Javy.
- Maskotka Javy – Duke (pamiątka po projekcie Green).

Czym jest Java?

- Java jest językiem zorientowanym obiektowo, który w dużej mierze opiera się na C i C++.
- Stanowi spójne logicznie środowisko programistyczne, posiadające najlepsze cechy swoich wzorców.
- W porównaniu z C i C++ Java jest zoptymalizowana. Usunięto mechanizmy będące przyczyną częstych błędów programistycznych, takie jak:
  - wskaźniki,
  - wielokrotne dziedziczenie,
  - przeciążanie operatorów.
- Początkowo Java służyła wyłącznie do tworzenia interaktywnego oprogramowania witryn internetowych. Z czasem obszar zastosowań Javy był systematycznie rozszerzany.

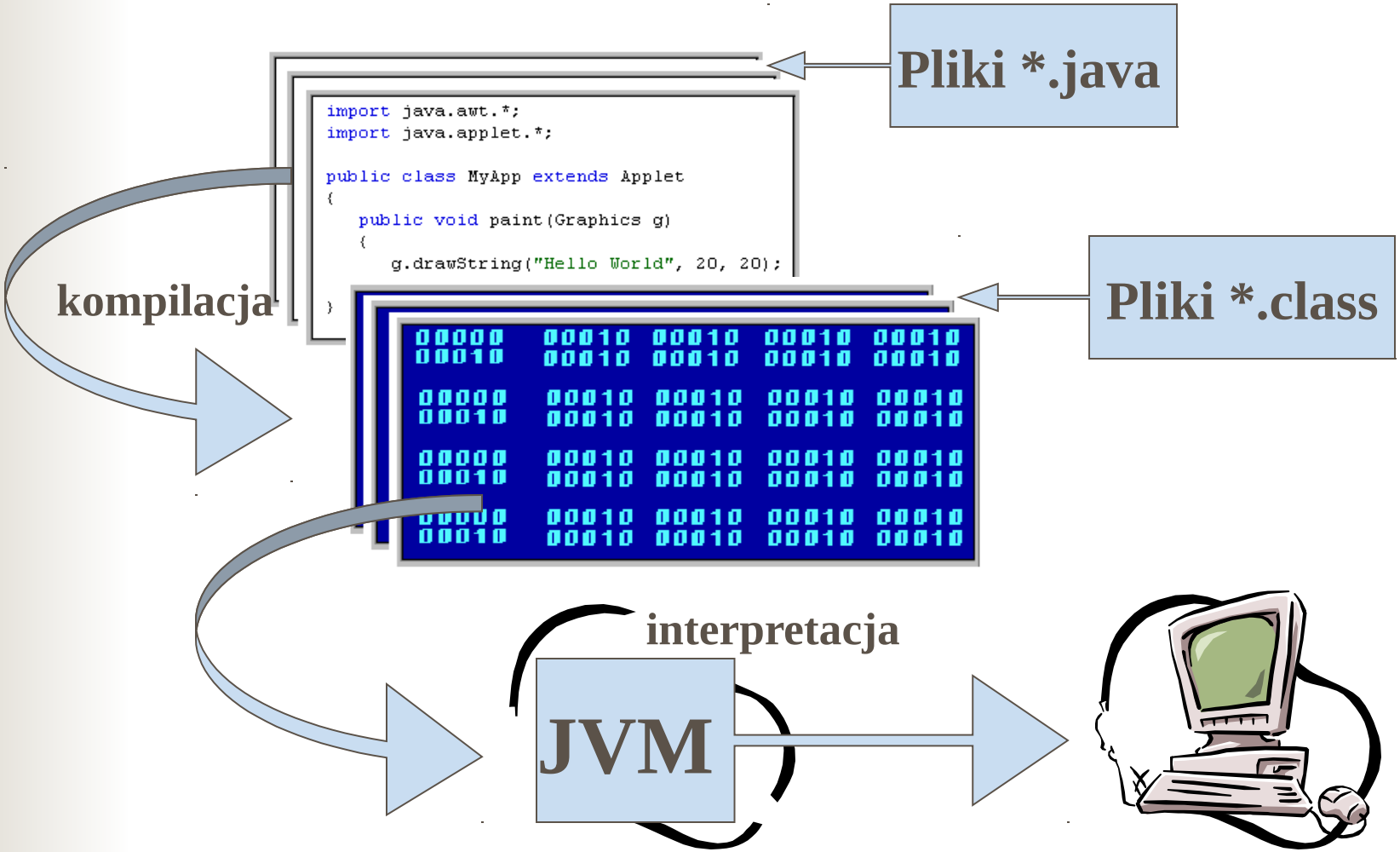
Cechy języka Java

- Niezależność od platformy sprzętowo-programowej.
  - Java to uniwersalny język programowania – raz napisany kod można wykorzystać w dowolnym środowisku, do którego przeniesiono JVM.
  - Uniwersalne środowisko programowania GUI i multimediów.
  - Uniwersalne środowisko dostępu do baz danych .
  - Uniwersalne środowisko programowania w sieci i w systemach rozproszonych.
- Możliwość budowania programów z gotowych komponentów.

Wirtualna maszyna Javy (JVM)

- **Java Virtual Machine to rodzaj wirtualnego komputera, który ma swój zestaw rejestrów, zestaw instrukcji, stos i pamięć dla programów.**
- **Dzięki standaryzacji maszyny wirtualnej, programy napisane w Javie są uniwersalne, tzn. wykonują się identycznie w każdym systemie operacyjnym.**
- **Programy napisane w Javie są kompilowane do poziomu kodu pośredniego, nazywanego kodem bajtowym Javy (*bytecode)*.**
- **Kod bajtowy jest interpretowany przez wirtualną maszynę JVM do postaci programu wykonywalnego dla danego systemu operacyjnego.**

Zasada działania



FUW

Narzędzia

- **Wszystkie narzędzia potrzebne do programowania w Javie znajdują się w bezpłatnym pakiecie SDK, który można pobrać z głównej witryny Javy**
  http://java.sun.com/j2se/
- **Kompletny kurs Javy, omawiający większość związanych z nią technologii zamieszczono na stronie**
  http://java.sun.com/docs/books/tutorial/

Edytory

Zintegrowane środowiska programistyczne Javy

Borland JBuilder – http://www.borland.com/products/download/

Eclipse – http://www.eclipse.org/

IBM VisualAge for Java – http://www7.software.ibm.com/vad.nsf

JCreator – http://www.jcreator.com/

Kawa – http://www.macromedia.com/

NetBeans – http://www.netbeans.org/

Sun Forte for Java – http://www.sun.com/forte/ffj/index.html

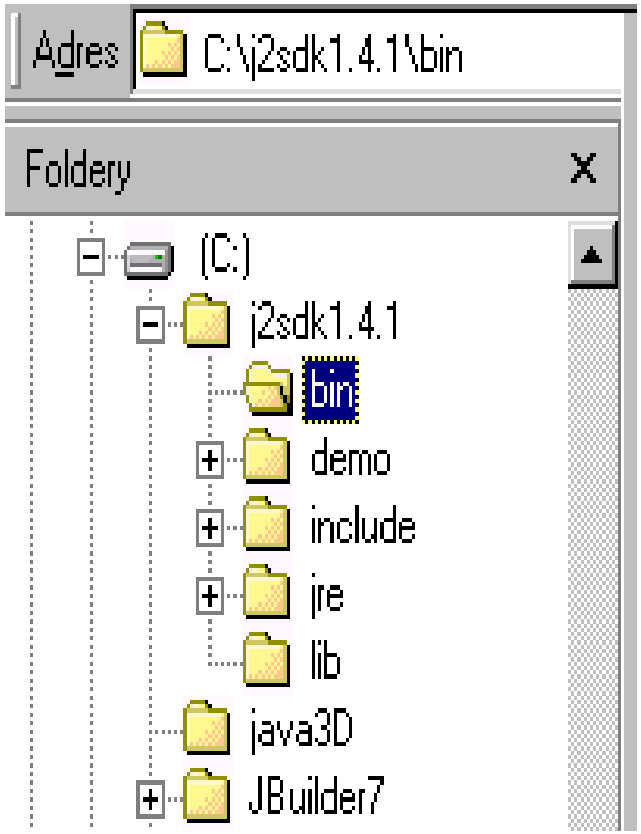
Sun One Studio – http://forte.sun.com/ffj/index.html

VIM – http://www.vim.org/

Zawartość pakietu SDK (JDK)

Po zainstalowaniu pakietu Java 2 SDK, SE w katalogu j2sdk1.4.1 zostanie umieszczony podkatalog bin zawierający szereg programów usługowych. Najważniejsze to:

- **javac – kompilator,**
- **java - interpreter,**
- **appletviewer – przeglądarka apletów,**
- **javadoc - generator dokumentacji,**
- **jdb – debuger,**
- **jar – narzędzie do tworzenia archiwów.**



Program w Javie

- Każdy program w Javie jest zestawem klas.
- Klasa jest podstawową jednostką enkapsulacji (nie można pisać kodu poza definicją klasy).
- Pisany przez nas program może być zapamiętany w jednym lub wielu plikach źródłowych o rozszerzeniu "**java**".
- Należy przestrzegać następującej konwencji dotyczącej nazewnictwa – nazwa klasy powinna być zgodna z nazwą pliku, który przechowuje program.

Aplkacje i aplety

- Wyróżniamy dwa rodzaje programów: aplikacje (**standalone programs**) i aplety (**applets**).
- Aplikacje mogą działać zarówno w trybie graficznym jak i tekstowym.
- Aplety działają jedynie w środowisku graficznym.
- Aby zobaczyć działanie aplikacji musimy mieć zainstalowaną w naszym komputerze wirtualną maszynę Javy – JVM.
- Aplety są wykonywane przez środowisko przeglądarek; są one widoczne wtedy, gdy przeglądarka posiada zintegrowaną wirtualną maszynę Javy.

Kompilacja

- Kompilator Javy wymaga, aby pliki źródłowe miały rozszerzenie "**java**".
- Pliki źródłowe są kompilowane za pomocą kompilatora Javy (javac.exe) do postaci kodu bajtowego (pośredniego), a nie kodu maszynowego.
- Polecenie kompilacji pliku źródłowego ma postać:

  javac nazwa_pliku.java
- Wynikiem kompilacji są pliki z rozszerzeniem "**class**", które mogą być wykonane przez maszynę wirtualną Javy.
- Podczas kompilacji pliku źródłowego każda klasa zostaje przeniesiona do swojego własnego pliku o nazwie właściwej zgodnej z nazwą klasy i rozszerzeniu "**class**".

Aplikacja

- Aby aplikacja mogła zostać uruchomiona, główna klasa musi zawierać metodę

  `public static void main(String args[])`

- Maszyna wirtualna Javy jest wywoływana za pomocą polecenia **java** z argumentami: nazwa pliku o rozszerzeniu "class" zawierającego metodę main() oraz argumenty wywołania tej metody, np.:

  `java nazwa_pliku arg1 arg2`

- Po załadowaniu klasy przez JVM sterowanie zostaje przekazane do metody main() i tu zaczyna się właściwe działanie programu: tworzenie obiektów, odwołania do innych klas aplikacji.

Aplet

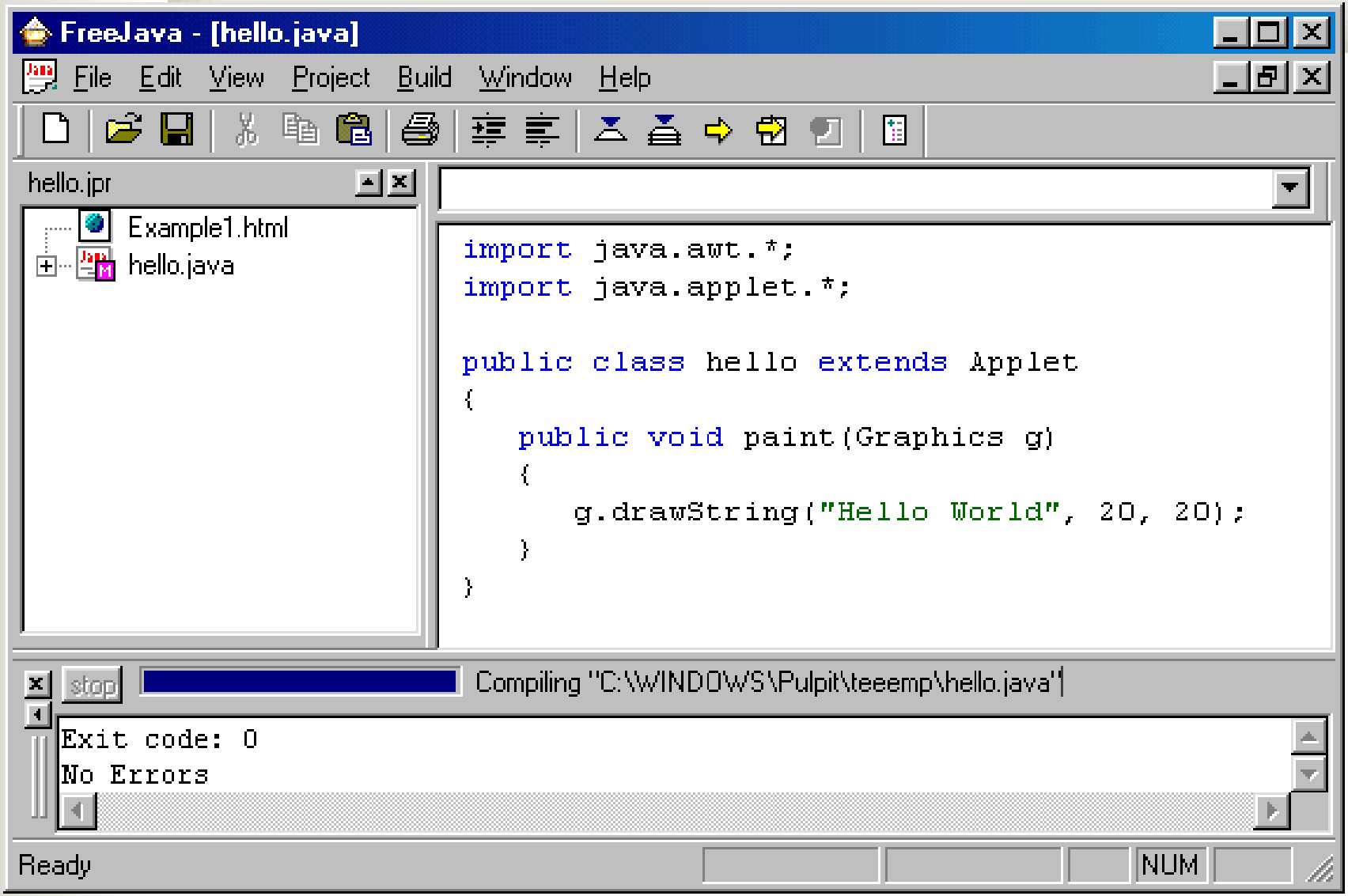
- Jedna z klas dziedziczy klasę Applet,
- Tworzymy plik HTML zawierający znacznik wywołania tej klasy, np.:
  - ```
    <applet code = ”MyApps.class”
     width = ”300” height = ”300”>
    </applet>
    ```
- Po napotkaniu tego znacznika przeglądarka ładuje plik `MyApps.class`, wywoływany jest konstruktor tej klasy, metoda inicjalizacyjna, itd.
    ```

Pisanie programu

Do pisania treści programu możemy użyć dowolnego edytora tekstu, pozwalającego na zapisanie tworzonego dokumentu jako zwykłego pliku tekstowego, np. Notatnika.

Będziemy korzystać z programu FreeJava – jest to proste IDE (zintegrowane środowisko programistyczne) pozwalające na bezpośrednią pracę z kodem źródłowym.

Okno programu jest podzielone na trzy części: okno projektu po lewej stronie pozwala na kontrolę poszczególnych klas, metod i plików, z prawej strony jest okno edytora kodu źródłowego, a na dole znajduje się okno komunikatów.


FreeJava - [hello.java]
File Edit View Project Build Window Help
hello.jpr
Example1.html
hello.java
import java.awt.*;
import java.applet.*;
public class hello extends Applet
{
public void paint(Graphics g)
{
g.drawString("Hello World", 20, 20);
}
}
stop
Compiling "C:\WINDOWS\Pulpit\teeemp\hello.java"
Exit code: 0
No Errors
Ready
NUM

Pierwszy program

Napiszemy teraz swoją pierwszą klasę zgodnie z przedstawionym niżej przykładem.

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

Kompilacja programu

- Utworzona przez nas klasa powinna być zapisana w pliku źródłowymo nazwie "Pierwsza.java" (istotna jest wielkość liter).
- W wyniku kompilacji powstanie plik z kodem bajtowym o nazwie "Pierwsza.class". Jest to już plik gotowy do wykonania przez JVM.
- Na koniec można uruchomić naszą aplikację, aby zobaczyć efekt jej działania.

```
Witamy
Exit code: 0
No Errors
```

FUW

Jak działa nasz program?

- Po załadowaniu klasy Pierwsza.class przez JVM sterowanie zostaje przekazane do metody `main()`.
- W naszym przykładzie metoda ta jest zdefiniowana z jednym parametrem; jest nim tablica args typu String. Dzięki temu możliwe jest przekazanie parametrów do aplikacji. Są one umieszczane w kolejnych elementach tej tablicy: args[0], args[1], args[2], itd.
  Liczbę przekazanych parametrów można uzyskać za pomocą metody `length()`, na przykład:

```
int liczbaParametrów = args.length();
```

- Metoda main() zawiera jedynie jedno polecenie `System.out.println();`

Pakiety

- Java dostarcza nam do dyspozycji tzw. pakiety - są to swoiste biblioteki klas, przy czym każda klasa w Javie należy do jakiegoś pakietu; zdefiniowana przez nas klasa należy do pakietu "bez nazwy" (domyślnego), definiowanego przez środowisko.
- Pakiety poza grupowaniem klas pełnią również rolę porządkującą i chronią przed kolizjami nazw. Jednym ze standardowych pakietów, nie wymagających deklaracji, jest pakiet java.lang, zawierający główne klasy języka Java.

Metoda println()

W klasie System pakietu java.lang zadeklarowano statyczne pole out związane ze standardowym wyjściem. Z polem tym związana jest metoda `println()`, która wypisuje na wyjściu wiersz podany jako argument. Czyli wynikiem działania programu będzie wypisanie na ekranie monitora pozdrowienia: "Witamy".

Modyfikacja programu

```
class Pierwsza
{
     public static void main (String args[])
     {
          pisz("Witamy");
     }
     public static void pisz (String s)
     {
          System.out.println(s);
     }
 }
```

Jaka jest różnica?

- Wynik działania naszej aplikacji jest dokładnie taki sam jak poprzednio, wywołanie w metodzie `main()` metody `pisz()` z parametrem `"Witamy"` oznacza polecenie wypisania podanego tekstu na ekranie monitora.
- Jedyna różnicą jest to, iż tym razem zdefiniowaliśmy w naszej klasie dwie metody - metodę główną sterującą działaniem aplikacji oraz metodę pomocniczą, służącą do wypisywania linii tekstu na ekranie.

Ćwiczenie 1

- **Do klasy Pierwsza dopisz funkcję rysującą trójkąt z gwiazdek. Trójkąt ma zostać utworzony za pomocą jednego wywołania polecenia System.out. println(). Oto nagłówek funkcji:**

  public static void rysuj_trojkat()

  Po uruchomieniu aplikacji na ekranie powinien być wyświetlony tekst:

```
Witamy
   *
  ***
 *****
```

Rozwiązanie

```
public static void main (String args[])
{
    pisz("Witamy");
    rysuj_trojkat();
}
public static void rysuj_trojkat()
{
    System.out.println(" *\n ***\n*****");
}
```

Ćwiczenie 2

- **Podobnie jak w ćwiczeniu pierwszym, do klasy Pierwsza dopisz funkcję wypisującą na ekranie napis**
  Ala
  /\/\ /\/
  i As
  Oto nagłówek funkcji:
  public static void pisz_tekst()

Rozwiązanie

```
System.out.println("Ala\n /\\/\\\n  i As");
```

Ćwiczenie 3

Do klasy Pierwsza dopisz funkcję rysującą n znaków "*"
w jednym wierszu , gdzie n jest parametrem funkcji.

Oto nagłówek funkcji:

```
public static void rysuj_gwiazdki( int n )
```

Spróbuj użyć tej funkcji dla różnych parametrów.

FUW

Rozwiązanie

```
public static void main (String args[])
{
    rysuj_gwiazdki(10);
}
public static void rysuj_gwiazdki(int n)
{
    for (int i=0; i<n; i++ )
    {
       System.out.print("*");
    }
    System.out.println();
}
```

Ćwiczenie 4

- **Napisz funkcję rysującą prostokąt o zadanych wymiarach, który składa się z samych gwiazdek.**
  Na przykład po wywołaniu tej funkcji z parametrami
  8 i 3 powinniśmy dostać:

```
* * * * * * * *
* * * * * * * *
* * * * * * * *
```

Oto nagłówek funkcji

```
public static void rysuj_prostokat( int a, int b )
```

Rozwiązanie

```
public static void rysuj_prostokat(int a, int b)
{
    for (int i=0; i<b; i++ )
    {
        for (int j=0; j<b; j++ )
        {
            System.out.print("*");
        }
        System.out.println();
     }
}
```

Ćwiczenie 5

- **Napisz funkcję rysującą trójkąt z gwiazdek. Wysokość trójkąta ma być podawana jako parametr. Na przykład po wywołaniu funkcji z parametrem 4 powinniśmy otrzymać następujący rysunek:**

```
*
* *
* * *
* * * *
```

Rozwiązanie

```
public static void rysuj_trojkat(int n)
{
    for (int i=0; i<n; i++ )
    {
        for (int j=0; j<=i; j++ )
        {
            System.out.print("* ");
        }
        System.out.println();
     }
}
```

Ćwiczenie 6

- **Zmodyfikuj program rysujący trójkąt tak, aby rysował trójkąty równoramienne. Na przykład po wywołaniu funkcji rysuj_trójkat z parametrem 4 powinniśmy otrzymać następujący rysunek:**

```
   *
  * * *
 * * * * *
* * * * * * *
```

Rozwiązanie

Trójkąt składa się z m wierszy, przy czym każdy kolejny wiersz jest dłuższy o dwie gwiazdki i ma mniejszą o jeden liczbę spacji poprzedzających. Wygodnie jest wprowadzić metody pomocnicze do rysowania spacji i gwiazdek.

```
public static void rysuj_spacje(int n)
{
   for (int i=0; i<n; i++ )
       System.out.print(" ");
 }
```

```
public static void rysuj_gwiazdki(int n)
{
   for (int i=0; i<n; i++ )
        System.out.print("* ");
}
public static void rysuj_trojkat(int m)
{
   for (int i=0; i < m; i++ )
   {
      rysuj_spacje(m-i-1);
      rysuj_gwiazdki(2*i+1);
      System.out.println();
   }
}
```