Język JAVA
podstawy programowania

Jacek Rumiński

wykład 2, część 1

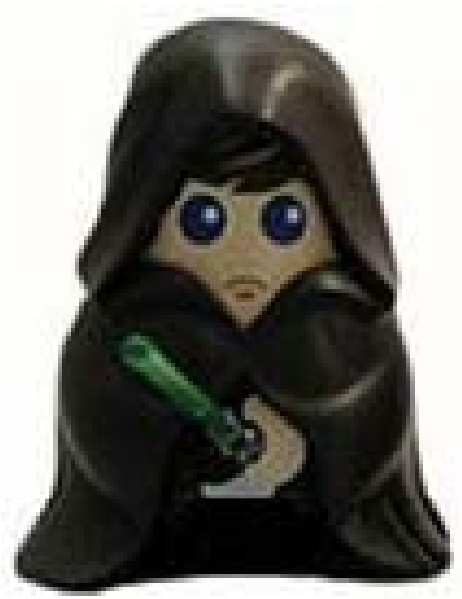
Plan wykładu:

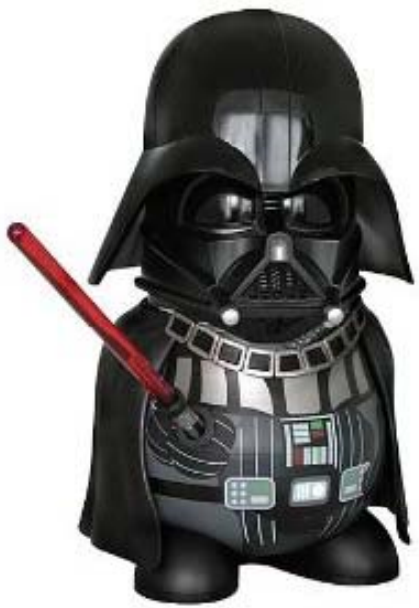
1. Rodzaje programów w Javie
2. Tworzenie aplikacji
3. Tworzenie apletów
4. Obsługa archiwów
5. Wyjątki
6. Klasa w klasie !

Jak korzystać z klas?

W czasie ostatniego wykładu podaliśmy sobie przykład klasy, opisującej (modelującej) rybę.

Dzisiaj rozpatrzmy bardzo podobny przykład klasy modelującej Rycerza Jedi !





źródło: www.play.com

Kod programu: RycerzJedi.java

```
public class RycerzJedi{

        //pola  - zmienne obiektu
        String nazwa;
        String kolor_miecza;

        //konstruktor
        RycerzJedi(String nazwa, String kolor_miecza){
                this.nazwa=nazwa;
                this.kolor_miecza=kolor_miecza;
        }

        //metody – funkcje obiektu
        void opis(){
                System.out.println("Rycerz "+nazwa+ " ma "+kolor_miecza+" miecz.");
        }

}// koniec class RycerzJedi
```

Co nowego w kodzie? - instrukcja wydruku w konsoli/terminalu

System.out.println("Rycerz "+nazwa+ " ma "+kolor_miecza+" miecz.");

System- standardowa klasa (szereg usług związanych z systemem/platformą)

out-obiekt, utworzony w momencie uruchamiania platformy Java, związany ze standardowym urządzeniem wyjścia (ekran). Obiekt ten jest zmienną klasy System. Każdy obiekt jest jakiegoś rodzaju, czyli jakiejś klasy. Obiekt „out" jest typu PrintStream. Każda klasa może mieć szereg zmiennych/pól oraz metod/funkcji. Klasa PrintStream (standardowa) ma szereg funkcji związanych z wysyłaniem danych, m.in. print (wyświetl w terminalu), println (wyświetl w terminalu i wstaw znak nowej linii) oraz printf (dla tych co znają język C).

Co nowego w kodzie? - instrukcja wydruku w konsoli/terminalu

"Rycerz "+nazwa+ " ma "+kolor_miecza+" miecz." – tekst, do którego dodajemy wartości zmiennych (m.in. *nazwa*). Znak dodawania (operator dodawania) jest przeciążony. Oznacza to, że może być użyty do dodania tekstu a nie tylko do (standardowo) dodawania liczb.

Zapamiętajmy – jeśli chcemy coś kontrolnie wyświetlić w konsoli:

System.out.println("cos");

No dobrze – mamy różne klasy i co dalej?

Otóż w Javie można utworzyć wiele różnych form programów, korzystając z utworzonych wcześniej klas.

Różne formy programów oznaczają tutaj docelowe środowiska, w których możliwe będzie uruchomienie programu w Javie.

Jakie są formy programów w Javie?

1. Tradycyjne aplikacje (J2SE)- programy uruchamiane na danym komputerze, pod kontrolą systemu operacyjnego, za pośrednictwem maszyny wirtualnej;
2. Aplet (J2SE) - programy uruchamiane w środowisku innego programu (np. przeglądarki, wywoływanie z „html-a”) – zwykle po stronie klienta;
3. Servlety (J2EE) – programy uruchamiane w środowisku serwera (tj. programu świadczącego usługi – SERVice providER);
4. Midlety (J2ME)– programy uruchamiane w środowisku urządzeń przenośnych (np. telefony komórkowe);
5. Inne - określone urządzenia, np. mote – sensore network node (Sun SPOT; dekodery (n-ka);
   karty chipowe (Java na karcie SIM);

Plan wykładu:

1. Rodzaje programów w Javie
2. Tworzenie aplikacji
3. Tworzenie apletów
4. Obsługa archiwów
5. Wyjątki
6. Klasa w klasie !

Jak utworzyć aplikację? Prosta zasada:

Aplikacja ma 1 punkt startowy: funkcja main() – maszyna wirtualna „wie” od jakiej instrukcji zacząć wykonywać nasz program !

```
public static void main(String args[]){
        //tu instrukcje naszego programu
}// koniec public static void main(String args[])
```

static- oznacza, że jest jedna taka sama funkcja dla wszystkich obiektów danej klasy (czyli jest to funkcja klasy – Class method)

args[] – oznacza tablicę argumentów wywołania naszego programu (args[0] – to argument 1; args[1] – to argument 2; itd.;)

args.length – rozmiar tablicy args, czyli liczba parametrów jakie podano przy wywołaniu programu – ***wniosek*** – tablica to też obiekt (bo ma cechy – length – rozmiar)

Jak utworzyć aplikację? Utwórz jedną klasę z funkcją main()!

Kod programu: Jedi.*java*

```
public class Jedi{

        public static void main(String args[]){
                System.out.println("Rycerz Luke ma niebieski miecz.");
        }// koniec public static void main(String args[])

}// koniec public class Jedi
```

„args” – może się nazywać inaczej (np. „a”, „arg”, „costam”) ważny jest typ ***String []*** – tablica z elementami typu **String** (ciąg znaków)
args[0] – to obiekt klasy String, można zatem wywołać różne funkcje, np. ***args[0].length()*** - nawiasy () – czyli **jest to metoda - zwraca** liczbę znaków pierwszego argumentu !
ALE ***args.length*** – **jest to pole** - rozmiar tablicy (nie ma nawiasów () oraz []).

A co z wykorzystaniem klas?

Dwa warianty:

1. Wszystko w jednym pliku (nie za dobrze – wersje rozwojowe). Tylko jedna klasa może być wówczas publiczna (oznaczona jako *public*) – i jest to główna klasa aplikacji (musi zawierać funkcję *main()*). Nazwa pliku – taka sama jak nazwa klasy publicznej !

2. Każda klasa w swoim pliku (tak lepiej – można wielokrotnie używać i rozwijać daną klasę).

PRZYKŁADY

Kod programu: ZlotJedi.java

```
class RycerzJedi{
        String nazwa;
        String kolor_miecza;
        RycerzJedi(String nazwa, String kolor_miecza){
                this.nazwa=nazwa;
                this.kolor_miecza=kolor_miecza;
        }
        void opis(){
                System.out.println("Rycerz "+nazwa+ " ma "+kolor_miecza+" miecz.");
        }
}// koniec class RycerzJedi
 public class ZlotJedi{
        public static void main(String args[]){
                RycerzJedi luke = new RycerzJedi("Luke", "zielony");
                RycerzJedi ben = new RycerzJedi("Obi-wan","niebieski");
                luke.opis();
                ben.opis();
        }// koniec public static void main(String args[])
}// koniec public class  ZlotJedi
```

Kod programu: RycerzJedi.java

```
//TAK JAK WCZEŚNIEJ
```

Kod programu: ZjazdJedi.java //w tym samym katalogu co RycerzJedi.java

```
public class ZjazdJedi{

        public static void main(String args[]){
                RycerzJedi luke = new RycerzJedi("Luke", "zielony");
                RycerzJedi ben = new RycerzJedi("Obi-wan","niebieski");
                luke.opis();
                ben.opis();
        }// koniec public static void main(String args[])

}// koniec public class  ZjazdJedi
```

Tak dużo lepiej – teraz wiele klas może korzystać z klasy RycerzJedi !!!

Pakiety – co to?

Klasy mogą być definiowane w ramach pakietu. Jak dotąd tego nie robiliśmy, stąd pakiet był domyślny.

Nazwy pakietów wprowadza się po to, aby grupować (logicznie, tematycznie, organizacyjnie) klasy. Pakiety mogą zawierać pakiety, itd. Przykładowo standardowe klasy Javy umieszczono w pakiecie o nazwie „java”. Podstawowe elementy języka w pakiecie „java.lang”, klasy związane z obsługą we/wy w pakiecie „java.io”, itd.

Pakiet (zbiór pakietów/klas) przechowywany jest jako katalog. Każdy zawarty pakiet to podkatalog. Każda klasa to plik NazwaKlasy.class (w kodzie źródłowym to NazwaKlasy.java).
Pakiety są wygodne – ale nieobowiązkowe !

Pakiety – co to?

Nazwy pakietów to często odwrócona nazwa dziedzinowa (adresów WWW). Przykładowo moje pakiety i klasy mógłbym przechowywać w pakiecie: pl.gda.biomed.jwr

Jak definiować pakiet – słowo kluczowe package a potem nazwa pakietu (wszystko w pierwszej linii kodu). Przykładowo:

```
package pl.gda.biomed.jwr;
```

Jak wskazywać pakiety, z których klas chcę skorzystać:

Słowo kluczowe import i nazwa pakietu wraz z klasą lub nazwa pakietu z *. Przykładowo

```
import java.io.*;  //wskazuję kompilatorowi, że będę korzystał z wielu klas
import java.net.Socket; //wskazuję, że będę używał klasy Socket z pakietu
```

Przejdźmy do demonstracji ->

Kod programu: RycerzJedi.java

```
package biomed.jwr.knights;
public class RycerzJedi{

        // Zawartość jak poprzednio, tylko pola, konstruktor i funkcja oznaczone jako public

}// koniec class RycerzJedi
```

Kod programu: ZebranieJedi.java

```
package biomed.jwr;
import biomed.jwr.knights.RycerzJedi; //lub import biomed.jwr.knights*;
public class ZebranieJedi{
        public static void main(String args[]){
                RycerzJedi luke = new RycerzJedi("Luke", "zielony");
                RycerzJedi ben = new RycerzJedi("Obi-wan","niebieski");
                luke.opis();
                ben.opis();
        }// koniec public static void main(String args[])
}// koniec public class  ZebranieJedi
```

Pakiety – jakie?

Wszystkie biblioteki JAVY – to pakiety z klasami.

Opis klasy i jej stosowania – w dokumentacji uporządkowanej według pakietów.

DEMO

Pakiety

Dla uproszczenia nauki – nie będziemy dalej definiować własnych pakietów.
Będziemy z nich jednak często korzystać !

Pamiętajmy, że biblioteki (pakiety) Javy to nie tylko to, co dostarcza Sun w dystrybucji Javy ale miliony pakietów (w większości za darmo) opracowanych przez programistów Javy na całym świecie (zarówno pracujących w znanych firmach, np. IBM, jak i wolnych strzelców).

Jeśli używamy jakąkolwiek klasę (poza wybranymi klasami tworzącymi podstawę języka) to zawsze musimy je wskazać w kodzie źródłowym – import ...

Plan wykładu:

1. Rodzaje programów w Javie
2. Tworzenie aplikacji
3. Tworzenie apletów (wykład 2, część 2)
4. Obsługa archiwów
5. Wyjątki
6. Klasa w klasie !