

PROGRAMOWANIE OBIEKTOWE I GUI

dr inż. Michał Tomaszewski

katedra Metod Programowania
Polsko-Japońska Akademia Technik Komputerowych

WYKŁAD 01

KLASY

Klasą zewnętrzną jest klasa **nie zawarta** w żadnej innej klasie.

Klasa zewnętrzna może być tylko **publiczna** albo **pakietowa**.

Klasą zewnętrzną jest klasa **nie zawarta** w żadnej innej klasie.

Klasa zewnętrzna może być tylko **publiczna** albo **pakietowa**.

```
package example01;
```

```
class Outer {  
    // ...  
}
```

Klasą wewnętrzną jest klasa zawarta w innej klasie, zadeklarowana bez specyfikatora **static** i zdefiniowana w miejscu, w którym mogłyby wystąpić składniki klasy zawierającej.

Klasą wewnętrzną jest klasa zawarta w innej klasie, zadeklarowana bez specyfikatora **static** i zdefiniowana w miejscu, w którym mogłyby wystąpić składniki klasy zawierającej.

- klasy wewnętrzne mogą być **prywatne**, **chroniona**, **publiczna** lub **pakietowa**

Klasą wewnętrzną jest klasa zawarta w innej klasie, zadeklarowana bez specyfikatora **static** i zdefiniowana w miejscu, w którym mogłyby wystąpić składniki klasy zawierającej.

- klasy wewnętrzne mogą być **prywatne**, **chroniona**, **publiczna** lub **pakietowa**
- klasa **wewnętrzna** nie może zawierać składników klasowych (w tym interfejsów)

KLASY WEWNĘTRZNE

Klasą wewnętrzną jest klasa zawarta w innej klasie, zadeklarowana bez specyfikatora **static** i zdefiniowana w miejscu, w którym mogłyby wystąpić składniki klasy zawierającej.

- klasy wewnętrzne mogą być **prywatne**, **chroniona**, **publiczna** lub **pakietowa**
- klasa **wewnętrzna** nie może zawierać składników klasowych (w tym interfejsów)

```
class Outer {  
    private int count = 10;  
  
    public  
    class Inner {  
        public int getCount(){  
            return count;  
        }  
    }  
}
```

Klasą **zanurzoną** jest klasa zawarta w innej klasie, zadeklarowana ze specyfikatorem **static** i zdefiniowana w miejscu, w którym mogłyby wystąpić składniki klasy zawierającej.

Klasą **zanurzoną** jest klasa zawarta w innej klasie, zadeklarowana ze specyfikatorem **static** i zdefiniowana w miejscu, w którym mogłyby wystąpić składniki klasy zawierającej.

- klasy **zanurzone** mogą być **prywatne**, **chroniona**, **publiczna** lub **paketowa**

Klasą **zanurzoną** jest klasa zawarta w innej klasie, zadeklarowana ze specyfikatorem **static** i zdefiniowana w miejscu, w którym mogłyby wystąpić składniki klasy zawierającej.

- klasy **zanurzone** mogą być **prywatne**, **chroniona**, **publiczna** lub **pakietowa**
- klasa **zanurzona** w odróżnieniu od klasy wewnętrznej, może zawierać składniki klasowe.

Klasą **zanurzoną** jest klasa zawarta w innej klasie, zadeklarowana ze specyfikatorem **static** i zdefiniowana w miejscu, w którym mogłyby wystąpić składniki klasy zawierającej.

```
class Outer {  
    private static int count = 10;  
  
    private  
    static class Immersed {  
  
        public static int count = 20;  
  
        public int getCount(){  
            return Outer.count+count;  
        }  
    }  
}
```

Klasą lokalną jest klasa zawarta w ciele funkcji.

Klasą lokalną jest klasa zawarta w ciele funkcji.

- klasy **lokalna** nie może być zadeklarowana ze specyfikatorem hermetyzacji, ani ze słowem **static**

Klasą lokalną jest klasa zawarta w ciele funkcji.

- klasy **lokalna** nie może być zadeklarowana ze specyfikatorem hermetyzacji, ani ze słowem **static**
- fabrykator obiekty tej klasy może wystąpić dopiero po jej definicji.

Klasą lokalną jest klasa zawarta w ciele funkcji.

- klasy **lokalna** nie może być zadeklarowana ze specyfikatorem hermetyzacji, ani ze słowem **static**
- fabrykator obiekty tej klasy może wystąpić dopiero po jej definicji.

```
package example01;

class Outer {

    public void fun(){
        class Local {
            // ...
        }
        Local loc = new Local();
    }
}
```

Klasą **anonimową** jest klasa niejawnie zdefiniowana tuż za fabrykatorem.

Klasą **anonimową** jest klasa niejawnie zdefiniowana tuż za fabrykatorem.

Definicja klasy anonimowej nie zawiera słowa kluczowego **class** ani fraz **extends** i **implements**

Klasą **anonimową** jest klasa niejawnie zdefiniowana tuż za fabrykatorem.

Definicja klasy anonimowej nie zawiera słowa kluczowego **class** ani fraz **extends** i **implements**

```
class Ex01 {  
  
    public void fun(){  
        new Object(){  
            public int val = 127;  
  
            public int getVal(){  
                return val;  
            }  
        }.getVal();  
    }  
}
```

KLASA ABSTRAKCYJNA

Klasa abstrakcyjna to taka, której nagłówek zawiera słowo kluczowe **abstract**.

Klasa abstrakcyjna to taka, której nagłówek zawiera słowo kluczowe **abstract**.

- klasa abstrakcyjna może, ale nie musi zawierać abstrakcyjnych metod;

Klasa abstrakcyjna to taka, której nagłówek zawiera słowo kluczowe **abstract**.

- klasa abstrakcyjna może, ale nie musi zawierać abstrakcyjnych metod;
- nie można tworzyć obiektów klasy abstrakcyjnej;

Klasa abstrakcyjna to taka, której nagłówek zawiera słowo kluczowe **abstract**.

- klasa abstrakcyjna może, ale nie musi zawierać abstrakcyjnych metod;
- nie można tworzyć obiektów klasy abstrakcyjnej;
- klasa abstrakcyjna może być klasą bazową dla innych klas.

Klasa abstrakcyjna to taka, której nagłówek zawiera słowo kluczowe **abstract**.

```
public
    abstract class Figure {

        private int width, height;

        public Figure(int width, int height){
            this.width = width;
            this.height = height;
        }
    }
```

Metoda abstrakcyjna to taka, której nagłówek zawiera słowo kluczowe `abstract`.

Metoda abstrakcyjna to taka, której nagłówek zawiera słowo kluczowe **abstract**.

Cechą metody abstrakcyjnej jest brak ciała metody.

Metoda abstrakcyjna to taka, której nagłówek zawiera słowo kluczowe **abstract**.

Cechą metody abstrakcyjnej jest brak ciała metody.

- jeżeli klasa zawiera metodę abstrakcyjną wówczas musi być klasą abstrakcyjną

Metoda abstrakcyjna to taka, której nagłówek zawiera słowo kluczowe **abstract**.

```
public
    abstract class Figure {

        private int width, hight;

        public Figure(int width, int hight){
            this.width = width;
            this.hight = hight;
        }

        public abstract void draw();
    }
```

DZIĘKUJĘ