Úvod do programovacích stylů

Přednáška 2. Objekty

verze z 27. září 2023

Jan Laštovička

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

Obsah

Opakování: bod

Opakování: bod

```
def make_point(x, y):
    return [x, y]

def get_point_x(point):
    return point[0]

def get_point_y(point):
    return point[1]

def set_point_x(point, x):
    point[0] = x

def set_point_y(point, y):
    point[1] = y
```

Opakování: bod

Opakování: bod

```
>>> point = make_point(3, 4)
>>> get_point_x(point)
3
>>> set_point_x(point, 5)
>>> get_point_x(point)
5
```

Data a kód, který s nimi pracuje, jsou oddělené.

Obsah

Objekt

Objekt

Data a kód, který s nimi pracuje, chápeme jako jeden celek nazývaný objekt.

Objekt

Data a kód, který s nimi pracuje, chápeme jako jeden celek nazývaný objekt.

Například: objekt `label` reprezentující popisek

Objekt

Data a kód, který s nimi pracuje, chápeme jako jeden celek nazývaný objekt.

Například: objekt `label` reprezentující popisek

Objekt je hodnota.

Objekt

Data a kód, který s nimi pracuje, chápeme jako jeden celek nazývaný objekt.

Například: objekt `label` reprezentující popisek

Objekt je hodnota.

```
>>> label
<omw.Label object at 0x11065fe50>
```

Objekt

Data a kód, který s nimi pracuje, chápeme jako jeden celek nazývaný objekt.

Například: objekt `label` reprezentující popisek

Objekt je hodnota.

```
>>> label
<omw.Label object at 0x11065fe50>
```

S objektem komunikujeme výhradně pomocí mechanizmu zasílání zpráv.

Zasílání zpráv

Zasílání zpráv

- *object* ... objekt
- *message* ... zpráva
- hodnoty: *value*, *arg1*, *arg2*, ...

```
object.message(arg1, arg2, ...) => value
```

- záslání zprávy *message* objektu *object* s argumenty *arg1*, *arg2*, ...

Zasílání zpráv

- *object* ... objekt
- *message* ... zpráva
- hodnoty: *value*, *arg1*, *arg2*, ...

```
object.message(arg1, arg2, ...) => value
```

- zaslání zprávy *message* objektu *object* s argumenty *arg1*, *arg2*, ...
- výsledkem je návratová hodnota *value*

Zasílání zpráv

- *object* ... objekt
- *message* ... zpráva
- hodnoty: *value*, *arg1*, *arg2*, ...

```
object.message(arg1, arg2, ...) => value
```

- záslání zprávy *message* objektu *object* s argumenty *arg1*, *arg2*, ...
- výsledkem je návratová hodnota *value*
- *object* ... příjemce zprávy

Zasílání zpráv

- *object* ... objekt
- *message* ... zpráva
- hodnoty: *value*, *arg1*, *arg2*, ...

```
object.message(arg1, arg2, ...) => value
```

- záslání zprávy *message* objektu *object* s argumenty *arg1*, *arg2*, ...
- výsledkem je návratová hodnota *value*
- *object* ... příjemce zprávy

Například:

```
>>> label.set_text("Pomeranč")
<omw.Label object at 0x11065fe50>
```

Zasílání zpráv

- *object* ... objekt
- *message* ... zpráva
- hodnoty: *value*, *arg1*, *arg2*, ...

```
object.message(arg1, arg2, ...) => value
```

- záslání zprávy *message* objektu *object* s argumenty *arg1*, *arg2*, ...
- výsledkem je návratová hodnota *value*
- *object* ... příjemce zprávy

Například:

```
>>> label.set_text("Pomeranč")
<omw.Label object at 0x11065fe50>
>>> label.get_text()
'Pomeranč'
```

Rozhraní

Rozhraní

Objektu můžeme zaslat jen některé zprávy.

Rozhraní

Objektu můžeme zaslat jen některé zprávy.

```
>>> label.get_text()
'Pomeranč'
>>> label.get_color()
AttributeError: 'Label' object has no attribute 'get_color'
```

Rozhraní

Objektu můžeme zaslat jen některé zprávy.

```
>>> label.get_text()
'Pomeranč'
>>> label.get_color()
AttributeError: 'Label' object has no attribute 'get_color'
```

objekt rozumí zprávě = lze mu ji zaslat

Například:

- `label` rozumí zprávě `get_text`
- `label` nerozumí zprávě `get_color`

Rozhraní

Objektu můžeme zaslat jen některé zprávy.

```
>>> label.get_text()
'Pomeranč'
>>> label.get_color()
AttributeError: 'Label' object has no attribute 'get_color'
```

objekt rozumí zprávě = lze mu ji zaslat

Například:

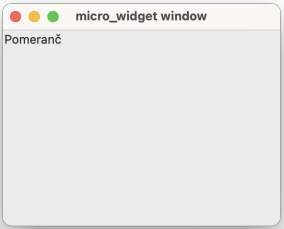
- `label` rozumí zprávě `get_text`
- `label` nerozumí zprávě `get_color`

rozhraní objektu = množina zpráv, kterým objekt rozumí

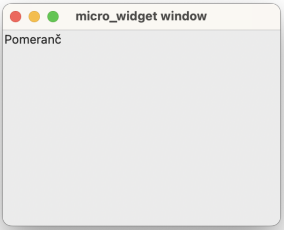
Například rozhraní objektu `label`:
{`get_text`, `set_text`, `get_x`, `set_x`, `get_y`, `set_y`, `move`}

Zasílání zpráv

Zasílání zpráv

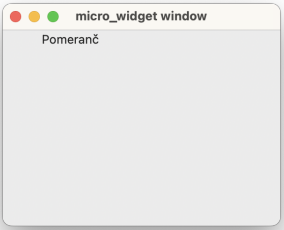


Zasílání zpráv



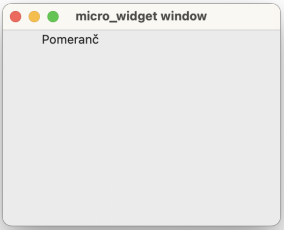

```
>>> label.get_x()
0
```

Zasílání zpráv




```
>>> label.get_x()
0
>>> label.set_x(40)
<omw.Label object at 0x11065fe50>
```

Zasílání zpráv




```
>>> label.get_x()
0
>>> label.set_x(40)
<omw.Label object at 0x11065fe50>
>>> label.get_x()
40
```

Vlastnosti

Vlastnosti

- data objektu

Například: `text`, `x`, `y`

Vlastnosti

- data objektu

Například: `text`, `x`, `y`

Vlastnost *property*:

- *object*`.get_`*property*`() => `*value*
  získání hodnoty vlastnosti
- *object*`.set_`*property*`(value) => `*object*
  nastavení hodnoty vlastnosti
  (pokud lze vlastnost nastavit)

Vlastnosti

- data objektu

Například: `text`, `x`, `y`

Vlastnost *property*:

- *object*`.get_`*property*`() => `*value*
  získání hodnoty vlastnosti
- *object*`.set_`*property*`(value) => `*object*
  nastavení hodnoty vlastnosti
  (pokud lze vlastnost nastavit)

Například vlastnost `text` objektu `label`:

- `label.get_text() => `*string*
- `label.set_text(`*string*`) => label`

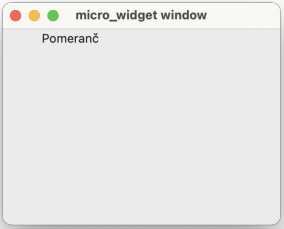
Posun

Posun

```
label.move(dx, dy) => label
```

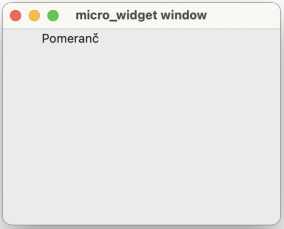
Posun

label.move(*dx*, *dy*) => *label*



Posun

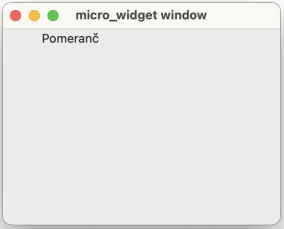
label.move(*dx*, *dy*) => *label*




```
>>>
```

Posun

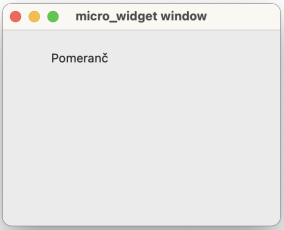
label.move(*dx*, *dy*) => *label*




```
>>> label.move(10, 20)
```

Posun

label.move(*dx*, *dy*) => *label*




```
>>> label.move(10, 20)
<omw.Label object at 0x11065fe50>
```

Zřetězení

Zřetězení

Pokud je to možné zaslání zprávy vrací příjemce.

Například: `set_`*`propery`*, `move`

Zřetězení

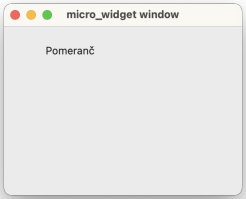
Pokud je to možné zaslání zprávy vrací příjemce.

Například: `set_`*`propery`*, `move`
Umožňuje zřetězení zasílání zpráv.

Zřetězení

Pokud je to možné zaslání zprávy vrací příjemce.

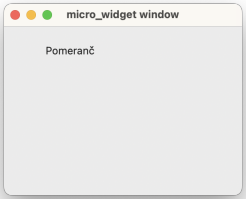
Například: `set_propery`, `move`
Umožňuje zřetězení zasílání zpráv.



Zřetězení

Pokud je to možné zaslání zprávy vrací příjemce.

Například: `set_propery`, `move`
Umožňuje zřetězení zasílání zpráv.

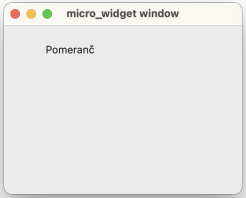



```
>>>
```

Zřetězení

Pokud je to možné zaslání zprávy vrací příjemce.

Například: `set_propery`, `move`
Umožňuje zřetězení zasílání zpráv.

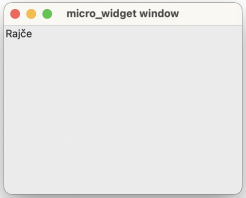



```
>>> label.set_text("Rajče").move(-50, -20)
```

Zřetězení

Pokud je to možné zaslání zprávy vrací příjemce.

Například: `set_propery`, `move`
Umožňuje zřetězení zasílání zpráv.

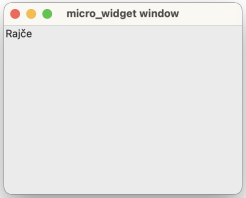



```
>>> label.set_text("Rajče").move(-50, -20)
<omw.Label object at 0x11065fe50>
```

Zřetězení

Pokud je to možné zaslání zprávy vrací příjemce.

Například: `set_propery`, `move`
Umožňuje zřetězení zasílání zpráv.




```
>>> label.set_text("Rajče").move(-50, -20)
<omw.Label object at 0x11065fe50>
```

Uzávorkování: `(label.set_text("Rajče")).move(-50, -20)`

Obsah

Třídy

Třídy

Strukturu a chování objektů určují třídy.

Třídy

Strukturu a chování objektů určují třídy.

- Třída definuje rozhraní objektu.

Třídy

Strukturu a chování objektů určují třídy.

- Třída definuje rozhraní objektu.
- Jména tříd začínají velkým písmenem.
  Například: `Label`

Třídy

Strukturu a chování objektů určují třídy.

- Třída definuje rozhraní objektu.
- Jména tříd začínají velkým písmenem.
  Například: `Label`
- Víceslovné názvy spojujeme velbloudí notací.
  Například: `CompoundWidget`

Třídy

Strukturu a chování objektů určují třídy.

- Třída definuje rozhraní objektu.
- Jména tříd začínají velkým písmenem.
  Například: `Label`
- Víceslovné názvy spojujeme velbloudí notací.
  Například: `CompoundWidget`

Každý objekt je přímou instancí nějaké třídy.

Třídy

Strukturu a chování objektů určují třídy.

- Třída definuje rozhraní objektu.
- Jména tříd začínají velkým písmenem.
  Například: `Label`
- Víceslovné názvy spojujeme velbloudí notací.
  Například: `CompoundWidget`

Každý objekt je přímou instancí nějaké třídy.

Například: `label` je přímou instancí třídy `Label`

Třídy

Strukturu a chování objektů určují třídy.

- Třída definuje rozhraní objektu.
- Jména tříd začínají velkým písmenem.
  Například: `Label`
- Víceslovné názvy spojujeme velbloudí notací.
  Například: `CompoundWidget`

Každý objekt je přímou instancí nějaké třídy.

Například: `label` je přímou instancí třídy `Label`

```
>>> label
<omw.Label object at 0x11065fe50>
```

Vytváření nových objektů

Vytváření nových objektů

```
Class() => object
```

- *Class* ... třída
- *object* je nová přímá instance třídy *Class*

Vytváření nových objektů

```
Class() => object
```

- *Class* ... třída
- *object* je nová přímá instance třídy *Class*

Například:

```
>>> label2 = Label()
>>> label2
<omw.Label object at 0x110490ad0>
>>> label
<omw.Label object at 0x11065fe50>
```

Obsah

Knihovna `omw`

Knihovna `omw`

- objektová nadstavba nad `micro_widget`

Knihovna `omw`

- objektová nadstavba nad `micro_widget` ... není potřeba vědět

Knihovna omw

- objektová nadstavba nad `micro_widget` ... není potřeba vědět
- manuál: `02_omw.pdf`

Knihovna `omw`

- objektová nadstavba nad `micro_widget` ... není potřeba vědět
- manuál: `02_omw.pdf`
- soubory: `micro_widget.py` a `omw.py`
- import: `from omw import *`

Knihovna `omw`

- objektová nadstavba nad `micro_widget` ... není potřeba vědět
- manuál: `02_omw.pdf`
- soubory: `micro_widget.py` a `omw.py`
- import: `from omw import *`
- nejlépe v IDLE

Knihovna omw

- objektová nadstavba nad `micro_widget` ... není potřeba vědět
- manuál: `02_omw.pdf`
- soubory: `micro_widget.py` a `omw.py`
- import: `from omw import *`
- nejlépe v IDLE

Třídy:

- `Window` (okno)
- `Label` (popisek)
- `Button` (tlačítko)
- `Entry` (textové pole)
- `Group` (skupina)

Okno

Okno

- třída `Window`
- vytvoření přímé instance zobrazí nové okno

Okno

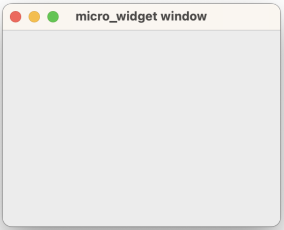
- třída `Window`
- vytvoření přímé instance zobrazí nové okno

```
>>> window2 = Window()
```

Okno

- třída `Window`
- vytvoření přímé instance zobrazí nové okno

```
>>> window2 = Window()
```



Vlastnost `widget`

Vlastnost `widget`

- určuje obsah okna

Vlastnost `widget`

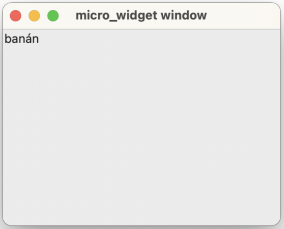
- určuje obsah okna

```
>>> label2.set_text("banán")
<omw.Label object at 0x110490ad0>
>>> window2.set_widget(label2)
<omw.Window object at 0x1106b3910>
```

Vlastnost `widget`

- určuje obsah okna

```
>>> label2.set_text("banán")
<omw.Label object at 0x110490ad0>
>>> window2.set_widget(label2)
<omw.Window object at 0x1106b3910>
```



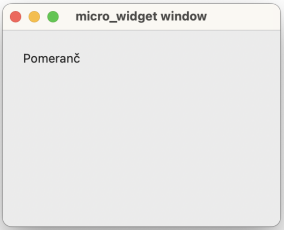
Celý program

Celý program

```
from omw import *
window = Window()
label = Label().set_text("Pomeranč").move(20, 20)
window.set_widget(label)
# window.main_loop()
```

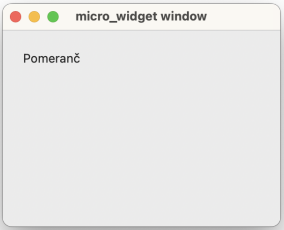
Celý program

```
from omw import *
window = Window()
label = Label().set_text("Pomeranč").move(20, 20)
window.set_widget(label)
# window.main_loop()
```



Celý program

```
from omw import *
window = Window()
label = Label().set_text("Pomeranč").move(20, 20)
window.set_widget(label)
# window.main_loop()
```



mimo IDLE: odkomentovat poslední řádek

Skupiny ovládacích prvků

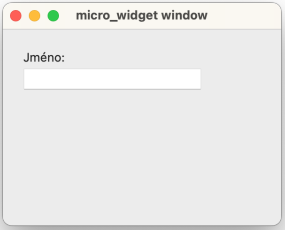
- Okno může obsahovat jen jeden ovládací prvek.
- Více prvků nutno dát do skupiny.
- instance třídy `Group`
- vlastnost `items`

Skupiny ovládacích prvků

- Okno může obsahovat jen jeden ovládací prvek.
- Více prvků nutno dát do skupiny.
- instance třídy `Group`
- vlastnost `items`

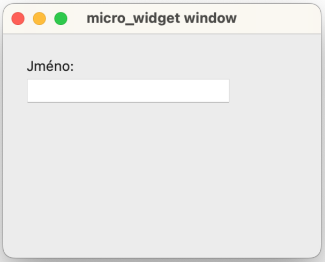
Například:

```
group = Group().set_items([label, entry])
window.set_widget(group)
```

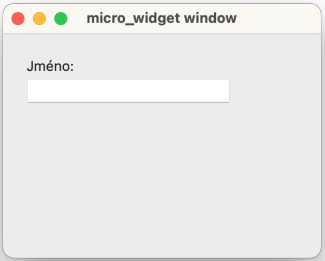


Posun

Posun



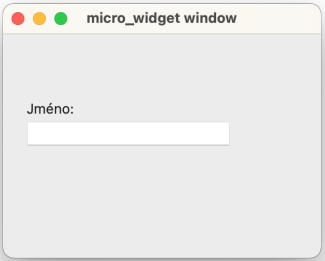
Posun



Skupina rozumí zprávě `move`:

```
>>> group.move(0, 40)
```

Posun



Skupina rozumí zprávě `move`:

```
>>> group.move(0, 40)
<omw.Group object at 0x108f40e10>
```

Rozdíl `micro_widget` a `omw`

Rozdíl `micro_widget` a `omw`

- prvky v `omw` lze vytvořit v libovolném pořadí
  Například: popisek před oknem

Rozdíl `micro_widget` a `omw`

- prvky v `omw` lze vytvořit v libovolném pořadí
  Například: popisek před oknem
- `omw` má lepší názvy akcí
  Například: `set_text` oproti `set_label_text`

Rozdíl `micro_widget` a `omw`

- prvky v `omw` lze vytvořit v libovolném pořadí
  Například: popisek před oknem
- `omw` má lepší názvy akcí
  Například: `set_text` oproti `set_label_text`
- ovládací prvky v `omw` nemají destruktor

Rozdíl `micro_widget` a `omw`

- prvky v `omw` lze vytvořit v libovolném pořadí
  Například: popisek před oknem
- `omw` má lepší názvy akcí
  Například: `set_text` oproti `set_label_text`
- ovládací prvky v `omw` nemají destruktor
- `omw` umožňuje shlukovat prvky

Rozdíl `micro_widget` a `omw`

- prvky v `omw` lze vytvořit v libovolném pořadí
  Například: popisek před oknem
- `omw` má lepší názvy akcí
  Například: `set_text` oproti `set_label_text`
- ovládací prvky v `omw` nemají destruktor
- `omw` umožňuje shlukovat prvky
- `omw` zatím nereaguje na uživatelský vstup