Programowanie obiektowe

Podstawy programowania w VBA

pojęcia podstawowe

obiektowy model aplikacji, podstawowe obiekty

procedury i funkcje użytkownika

operacje arytmetyczne, zmienne i stałe

Literatura

- Alexander M., Kusleika R. *Excel 2016 PL. Programowanie w VBA,* Helion, Gliwice 2016
- Kuciński K., *Visual Basic dla Excela w przykładach*, Wydawnictwo Witanet 2015
- Lewandowski M., *VBA dla Excela 2010. Leksykon kieszonkowy*, Helion, Gliwice 2012
- Lewandowski M., *Tworzenie makr w VBA dla Excela 2010/2013 Ćwiczenia*, Helion, Gliwice 2014
- McFedries P., *Microsoft Office 2007 PL język VBA i makra: usprawnij działanie najpopularniejszego pakietu biurowego,* Helion, Gliwice 2008
- Walkenbach J., *Excel 2013 PL. Programowanie w VBA dla bystrzaków*, Helion, Gliwice 2014

- Walkenbach J., *Excel 2016 PL. Biblia - Helion*, Gliwice 2016
- Wrotek W., *VBA dla Excela 2016 PL: 222 praktyczne przykłady,* Helion, Gliwice 2016
- Baca J., *Excel 2016 i programowanie VBA. Kurs video. Poziom drugi. Zaawansowane techniki tworzenia makr*, Videopoint 2016
- Jelen B., Syrstad T., *Excel 2016 VBA i makra*, PROMISE 2016.

Pojęcia podstawowe

Algorytm – przepis postępowania prowadzący do rozwiązania określonego zadania; zbiór poleceń określających sposób przetwarzania zbioru danych ze wskazaniem kolejności w jakiej mogą być wykonane.

Język programowania – sformalizowany język opisu algorytmów przeznaczonych do wykonywania przez komputer.

Język wysokiego poziomu – niezwiązany z określonym typem komputera, wykonanie programu wymaga wcześniejszego przetłumaczenia przy pomocy odpowiedniego translatora.

Programowanie – konstruowanie programów i przygotowanie ich do eksploatacji; kodowanie algorytmów w danym języku programowania.

Program – algorytm zapisany w określonym języku programowania wraz ze strukturami danych na których operuje. Program jest przepisem wyrażonym w odpowiedniej notacji według którego komputer lub inne urządzenie interpretujące wykonuje czynności przewidziane w algorytmie.

Zastosowanie VBA w programie Excel

- Rejestrowanie makr automatyzujących powtarzalne czynności.
- Tworzenie makr realizujących złożone czynności, wymagające komunikacji z użytkownikiem.
- Tworzenie nowych funkcji arkuszowych, niedostępnych w programie Excel, które można wykorzystać w formułach.
- Tworzenie przycisków i innych elementów sterujących dostępnych w arkuszach.
- Tworzenie nowych funkcji aplikacji realizujących operacje, które nie są niedostępne w programie Excel.
- Tworzenie elementów interfejsu użytkownika (okna dialogowe).
- Tworzenie dodatków rozszerzających możliwości aplikacji.

Programowanie obiektowe

Programowanie obiektowe (***OOP****, object-oriented programming*) – programowanie, zazwyczaj w obiektowym języku programowania, które polega na wyodrębnieniu obiektów, ich własności oraz metod. Każdy obiekt traktowany jest jako samodzielny, zamknięty fragment kodu, a funkcjonowanie programu opiera się na współpracy pomiędzy obiektami.

Obiekt – abstrakcyjny byt reprezentujący pewną rzecz lub pojęcie obserwowane w modelowanym systemie. Obiekt jest odróżnialny od innych obiektów, ma nazwę i dobrze określone granice.

Własność (atrybut, pole) – istotna z punktu widzenia modelu cecha obiektu.

Metoda – operacja przypisana do obiektu, operacja, którą potrafi wykonać dany obiekt.

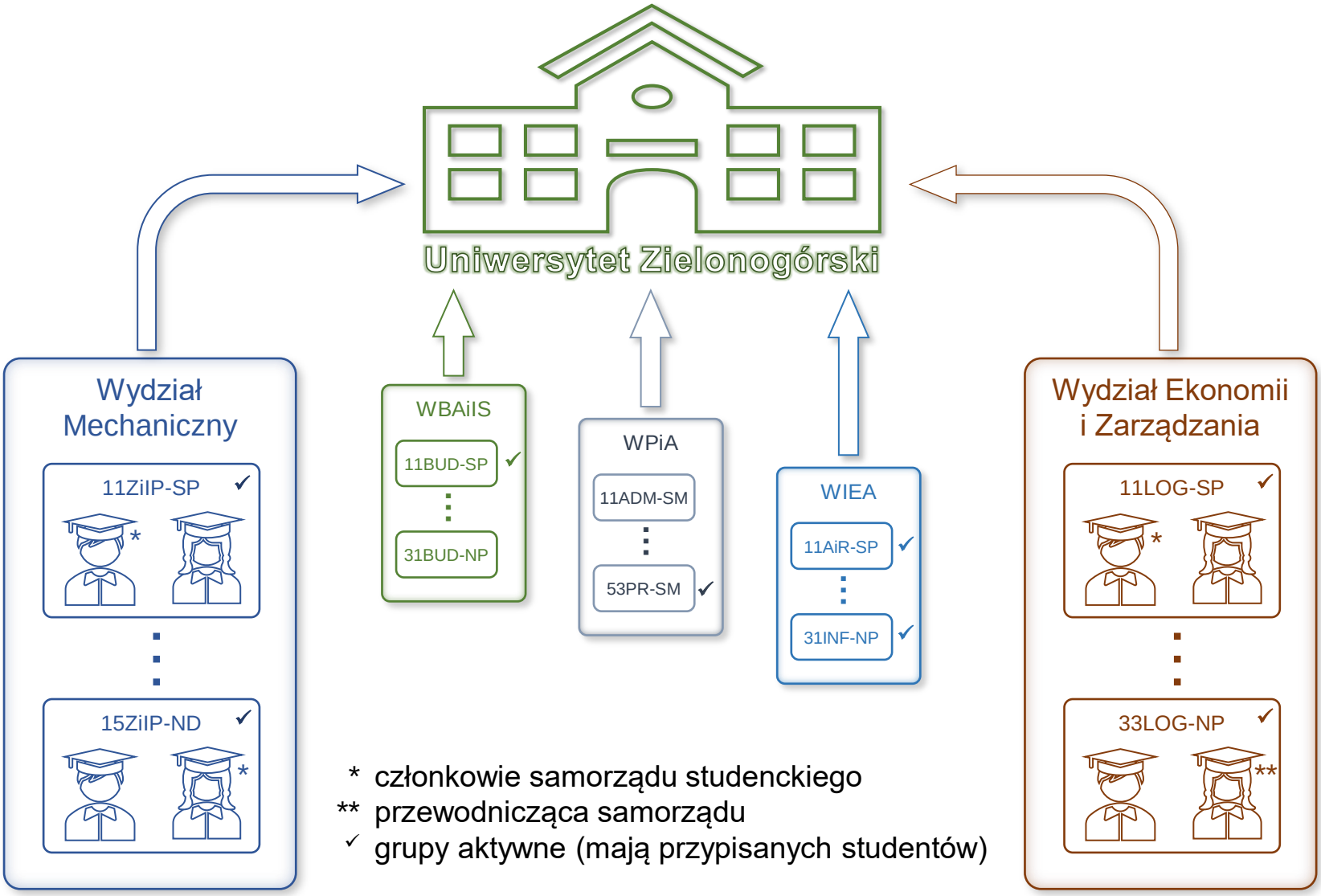
Klasa – definicja struktury pewnej grupy obiektów.

Przykłady

Obiekt: *Jan Kowalski*, klasa: *Student*
Własności: nazwisko, imię, data urodzenia, grupa, lista ocen.
Metody: oblicz wiek, oblicz średnią, napisz kolokwium.

Obiekt: *Uniwersytet Zielonogórski*, klasa: *Instytucja*
Własności: adres, lista wydziałów, lista studentów.
Metody: przyjmij studenta, skreśl studenta z listy, wydaj dyplom.

Obiektowy model uczelni



Obiektowy model uczelni

Klasy obiektów

- Uniwersytet (własności: nazwa, rektor, lista wydziałów, samorząd studencki, itp.),
- Wydział (własności: nazwa, dziekan, lista grup, itp.),
- Grupa studencka (własności: nazwa, wydział, lista osób, starosta, itp.)
- Student (własności: nazwisko, imię, data urodzenia, grupa studencka, itp.).

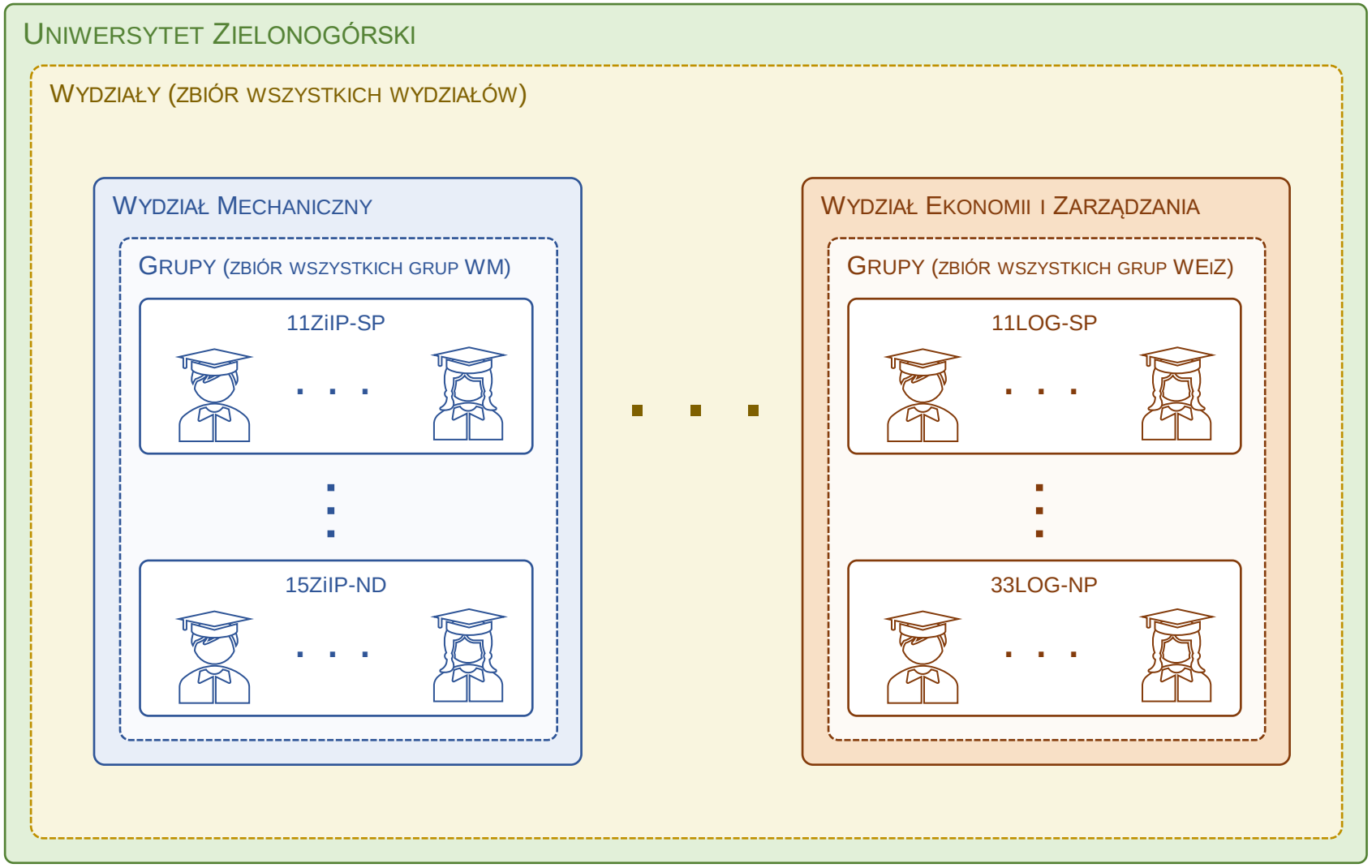
Obiekty

- Uniwersytet Zielonogórski
- Wydziały: Mechaniczny, Ekonomii i Zarządzania, Prawa i Administracji, …
- Grupy: 21ZiIP-SP, 21ZiIP-NP, 11ZiIP-SD, 15ZiIP-SD, 11ZiIP-ND, 15ZiIP-ND,…
- Studenci z poszczególnych grup (każda osoba stanowi odrębny obiekt).

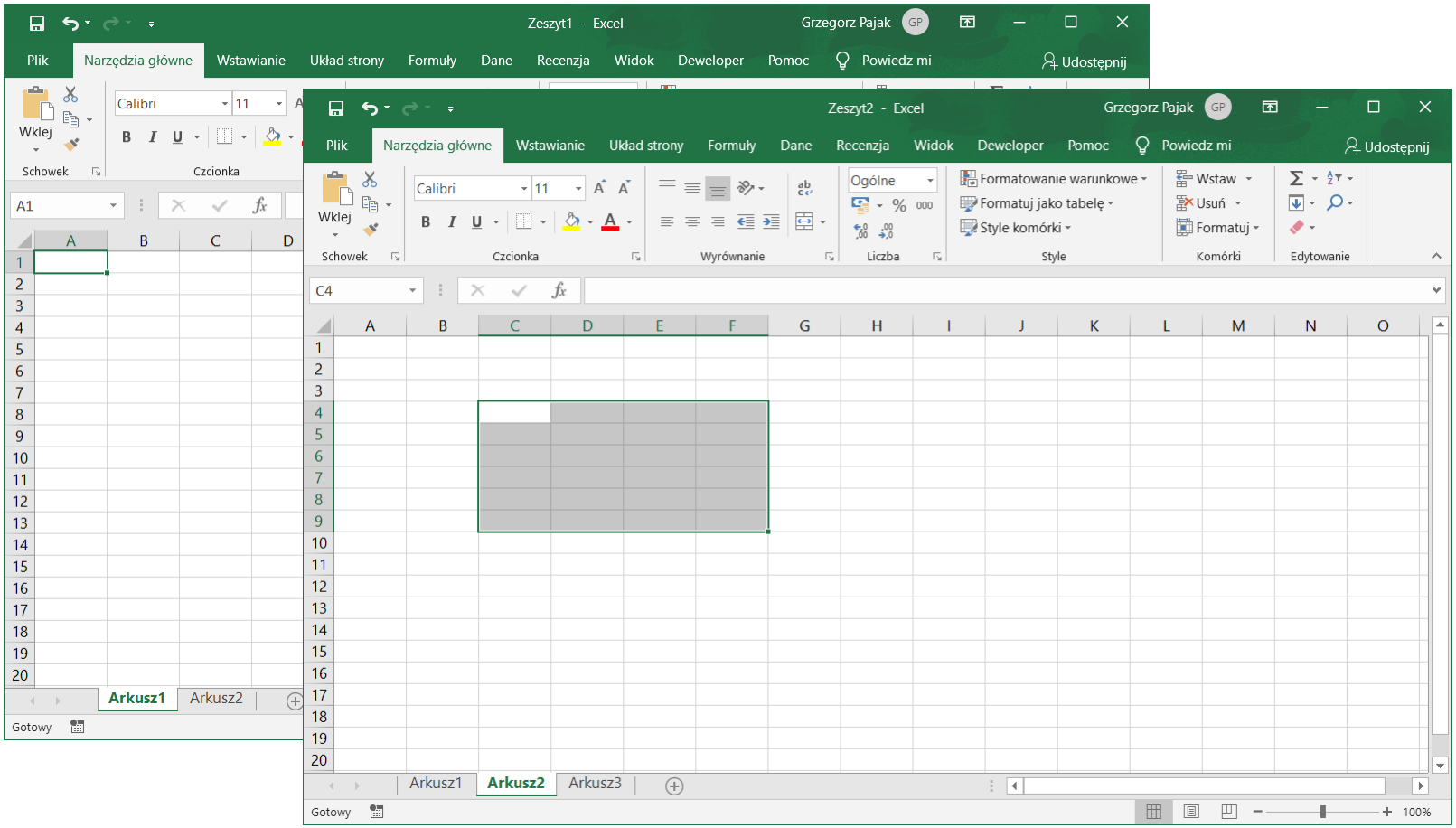
Zbiory obiektów (kolekcje)

- Zbiór wydziałów,
- Zbiór grup studenckich na wydziale,
- Zbiór studentów w grupie.

Obiektowy model uczelni



Obiektowy model Excel-a



Dwa dokumenty: *Zeszyt1*, *Zeszyt2*; dokument aktywny: *Zeszyt2*
Dokument *Zeszyt1*: dwa arkusze: *Arkusz1*, *Arkusz2*
Dokument *Zeszyt2*: trzy arkusze: *Arkusz1*, *Arkusz2*, *Arkusz3*; arkusz aktywny: *Arkusz2*
Aktywna komórka w *Arkusz2*: *C4*, wybrany zakres w *Arkusz2*: *C4:F9*

Obiektowy model Excel-a

Klasy obiektów

- Application (własności: użytkownik, aktywna komórka, wybrany zakres, itp.),
- Workbook (własności: nazwa dokumentu, lista arkuszy, arkusz aktywny, itp.),
- Worksheet (własności: nazwa arkusza, zbiór komórek, używany zakres komórek, itp.),
- Range (własności: adres, komórki, kolumny, wiersze, wprowadzone wartości, itp.).

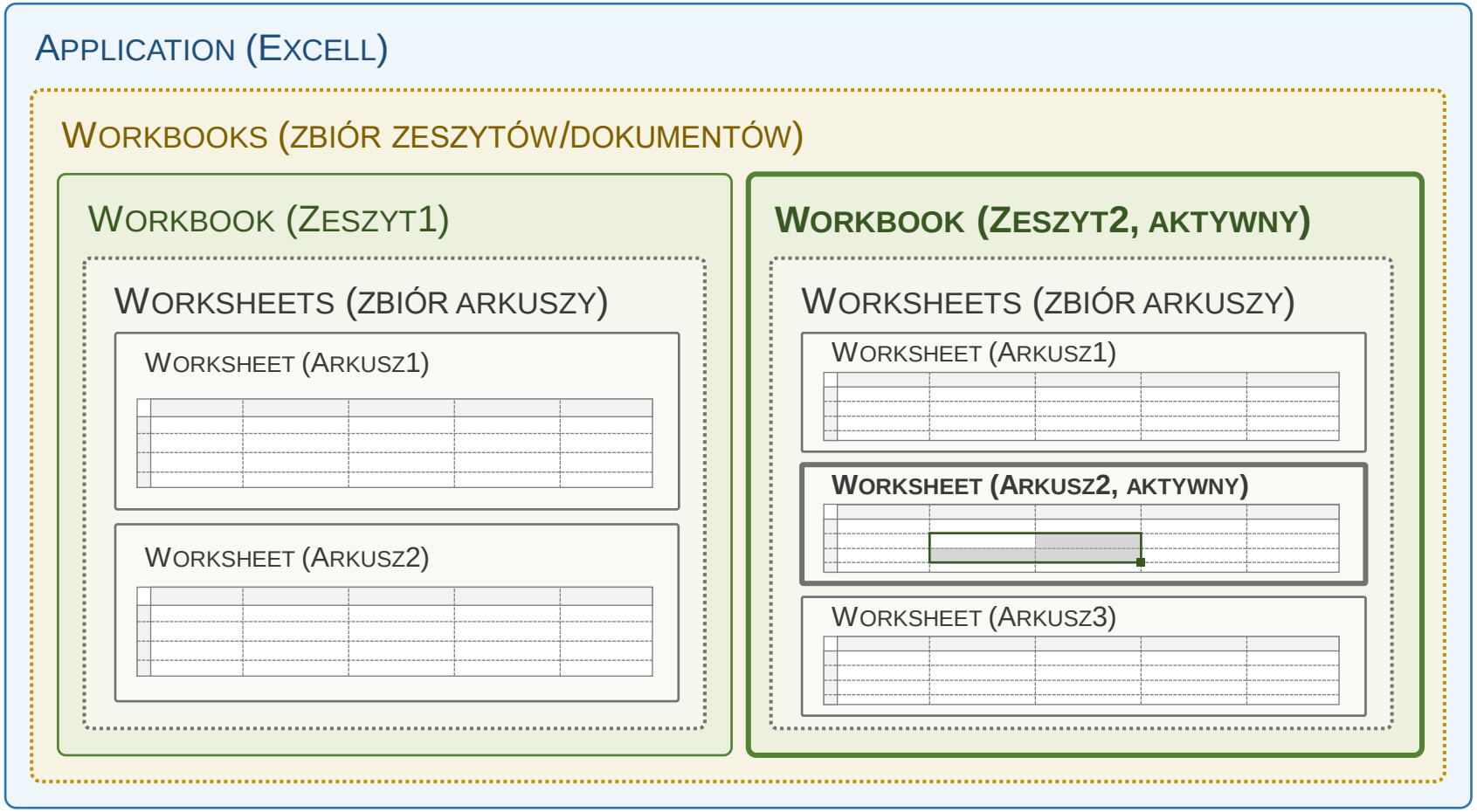
Obiekty

- Application – aplikacja (Excel),
- Workbook – pojedynczy dokument (*Zeszyt1* i *Zeszyt2*),
- Worksheet – pojedynczy arkusz (*Arkusz1-2* w *Zeszyt1* i *Arkusz1-3* w *Zeszyt2*),
- Range – pojedyncze komórki arkuszy, zakresy komórek.

Zbiory obiektów (kolekcje)

- Workbooks – zbiór wszystkich otwartych dokumentów,
- Worksheets – zbiór wszystkich arkuszy danego dokumentu,
- Range – zbiór komórek (np. w zaznaczonym zakresie, w kolumnie lub wierszu).

Obiektowy model Excel-a



Uwaga: powyższy model jest niekompletny i zawiera pewne uproszczenia. Odpowiada przykładowi przedstawionemu na s.1-9.

Obiekt Application

Application reprezentuje całą aplikację, istnieje od chwili uruchomienia Excel-a, zawsze występuje tylko w jednym egzemplarzu.

Wybrane własności

- **ActiveCell** – klasa Range, aktywna komórka (może być tylko jedna),
- **ActiveSheet** – klasa Worksheet, aktywny arkusz (Worksheet),
- **ActiveWorkbook** – klasa Workbook, aktywny dokument (Workbook),
- **Selection** – klasa Range, aktualnie wybrany zakres komórek,
- **UserName** – string, nazwa użytkownika,
- **Worksheets** – klasa Worksheets, kolekcja arkuszy aktualnego dokumentu.

Wybrane metody

- **Calculate** – przelicza wszystkie otwarte dokumenty (Workbook),
- **InputBox(***text***)** – wyświetla okno z umożliwiające wprowadzenie wartości,
- **Quit** – zamyka program.

Kolekcja Workbooks

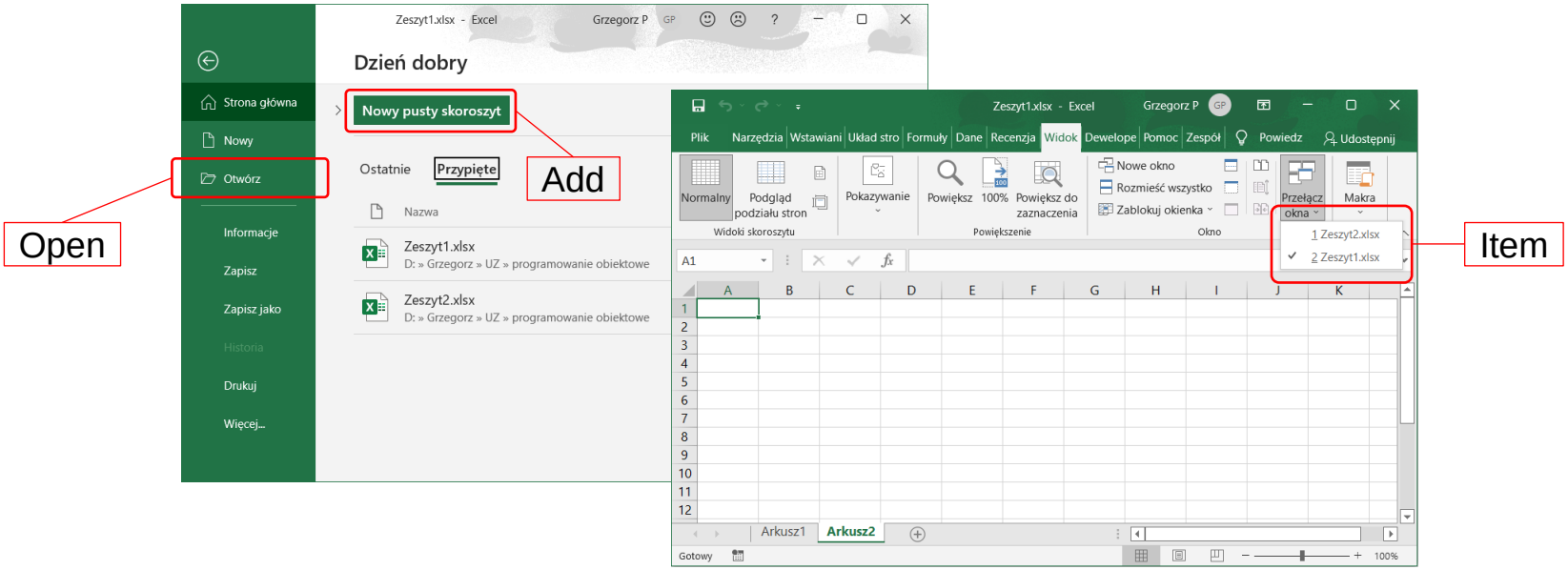
Workbooks reprezentuje kolekcję aktualnie otwartych dokumentów.

Wybrane własności

- **Count** – Long, liczba otwartych dokumentów (elementów kolekcji),
- **Item(***nazwa*|*index***)** – klasa Workbook, dokumentu o określonej nazwie lub indeksie.

Wybrane metody

- **Add** – dodaje nowy, pusty dokument (Workbook) do kolekcji,
- **Open(***nazwa***)** – otwiera dokument o określonej *nazwie*.



Obiekt Workbook

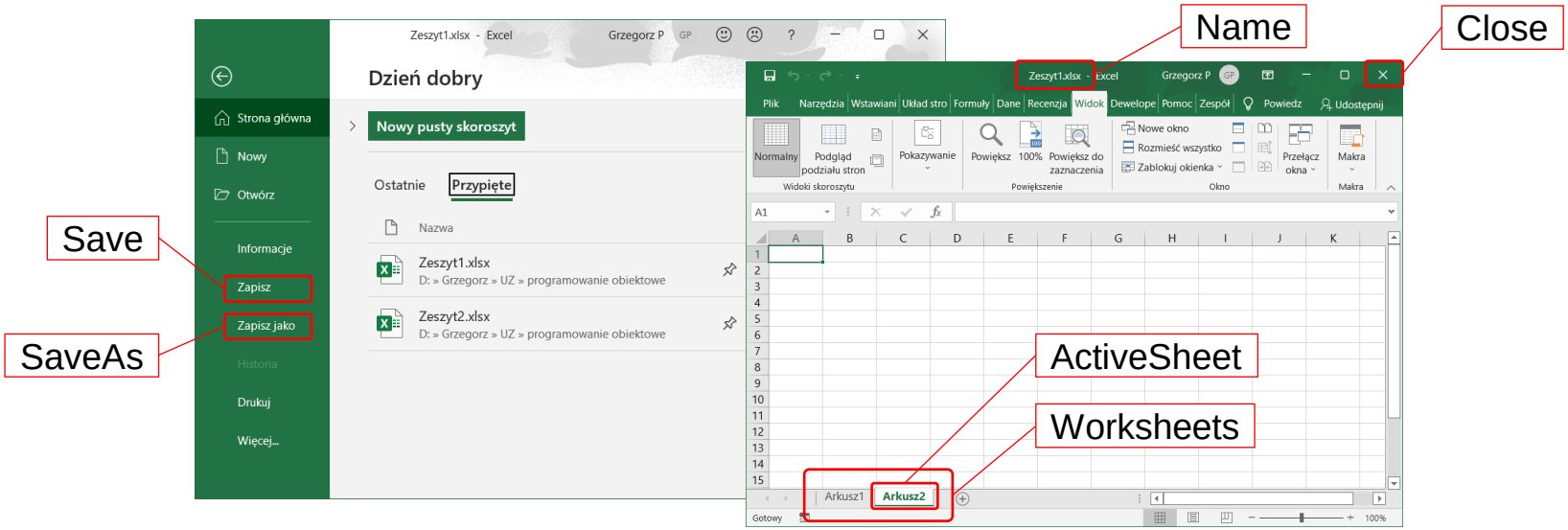
Workbook reprezentuje pojedynczy dokument w kolekcji Workbooks.

Wybrane własności

- **ActiveSheet** – klasa Worksheet lub Chart, aktywny arkusz lub wykres,
- **Name, Path, FullName** – String, kolejno: nazwa, ścieżka, pełna nazwa dokumentu,
- **Worksheets** – klasa Worksheets, kolekcja arkuszy tworzących dokument.

Wybrane metody

- **Activate** – aktywuje Woorkbook,
- **Close** – zamyka Workbook,
- **Save**, **SaveAs(***nazwa***)** – zapisuje Workbook pod dotychczasową lub nową *nazwą*.



Kolekcja Worksheets

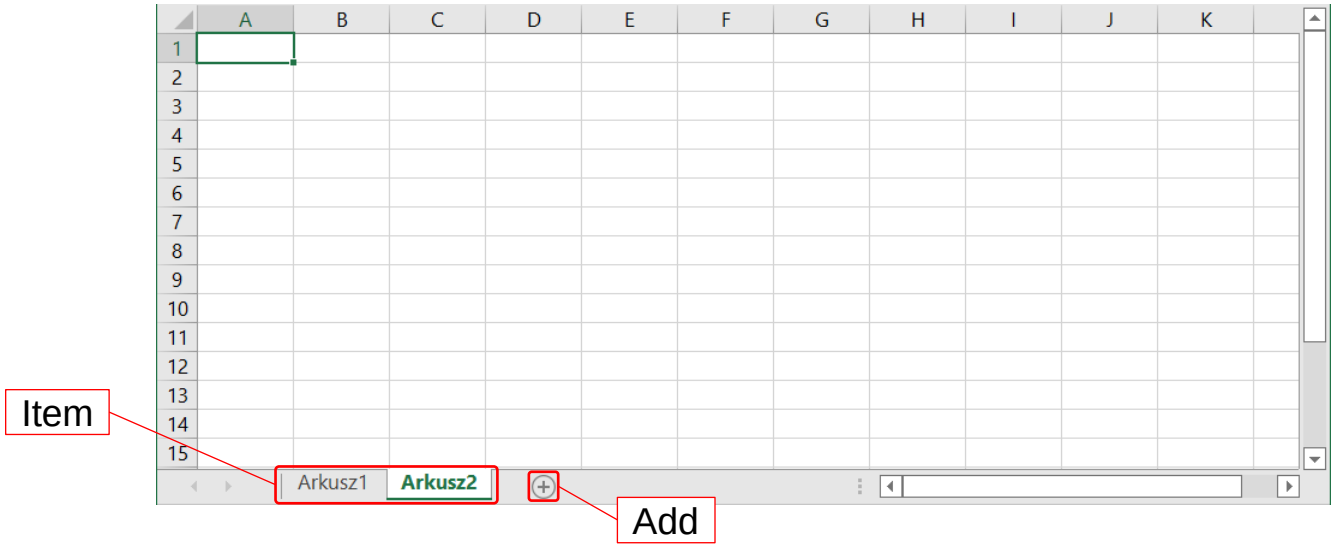
Worksheets reprezentuje kolekcję arkuszy lub wykresów.

Wybrane własności

- **Count** – Long, liczba elementów kolekcji,
- **Item(***nazwa*|*index***)** – klasa Worksheet, element o określonej nazwie lub indeksie.

Wybrane metody

- **Add(***przed*, *po***)** – dodaje nowy, pusty arkusz (Worksheet) do kolekcji *przed* lub *po* arkuszu istniejącym.



Obiekt Worksheet

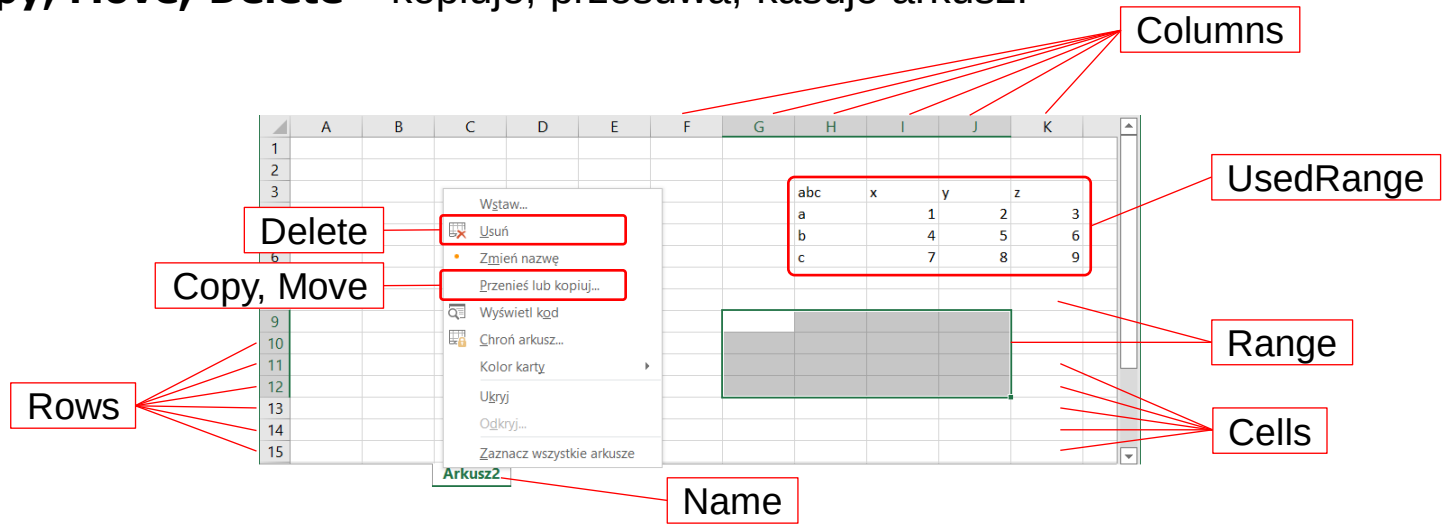
Worksheet reprezentuje pojedynczy arkusz w kolekcji Worksheets

Wybrane własności

- **Cells(***indeks***)** – klasa Range, kolekcja komórek arkusza,
- **Columns, Rows** – klasa Range, kolekcje kolumn i wierszy,
- **Name**, **Index** – nazwa (String) lub numer (Long) arkusza w kolekcji,
- **Range(***zakres***)** – klasa Range, kolekcja komórek z wybranego zakresu,
- **UsedRange** – klasa Range, używany zakres arkusza.

Wybrane metody

- **Activate** – aktywuje (wybiera) arkusz,
- **Copy, Move, Delete** – kopiuje, przesuwa, kasuje arkusz.



Obiekt Range – własności

Range reprezentuje pewien zakres komórek (cały arkusz, wybrane kolumny, wybrane wiersze, wskazany zakres, itp.).

Wybrane własności

- **Address** – String, adres zakresu,
- **Cells** – klasa Range, kolekcja komórek zakresu,
- **Columns, Rows** – klasa Range, kolekcje kolumn i wierszy w zakresie,
- **Count, Column, Row** – Long, liczba komórek, numer pierwszej kolumny i wiersza,
- **End(***kierunek***)** – komórka z początku/końca zakresu (xlDown, xlToLeft, xlToRight, xlUp),
- **EntireRow** – klasa Range, cały wiersz (wiersze) zawierające zakres,
- **Font** – czcionka używana w zakresie (obiekt, zawiera nazwę, kolor, rozmiar, itd.),
- **Formula**, **FormulaR1C1** – formuła zawarta w komórce w notacji A1 lub R1C1,
- **Offset(***w,k***)** – klasa Range, zakres przesunięty o *w* wierszy i *k* kolumn,
- **Resize(***w, k***)** – klasa Range, zakres o *w* wierszach i *k* kolumnach,
- **Text** – String, zawartość zakresu w postaci tekstowej uwzględniającej formatowanie,
- **Value** – Variant, wartość wprowadzona w zakresie.

Uwaga: własności Formula..., Text i Value zwracają wartość pustą w przypadku zakresu obejmującego wiele komórek.

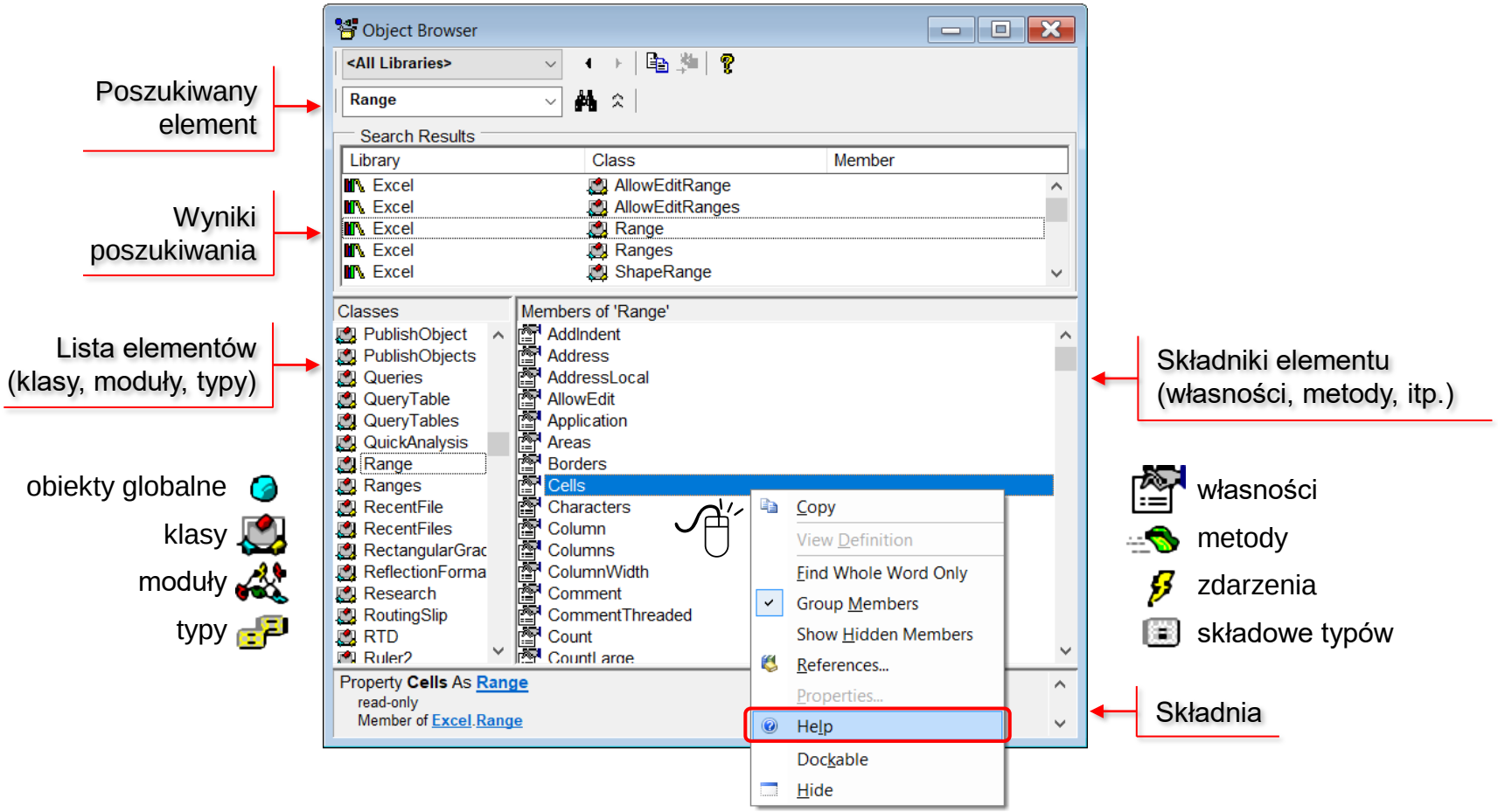
Obiekt Range – metody

Range reprezentuje pewien zakres komórek (cały arkusz, wybrane kolumny, wybrane wiersze, wskazany zakres, itp.).

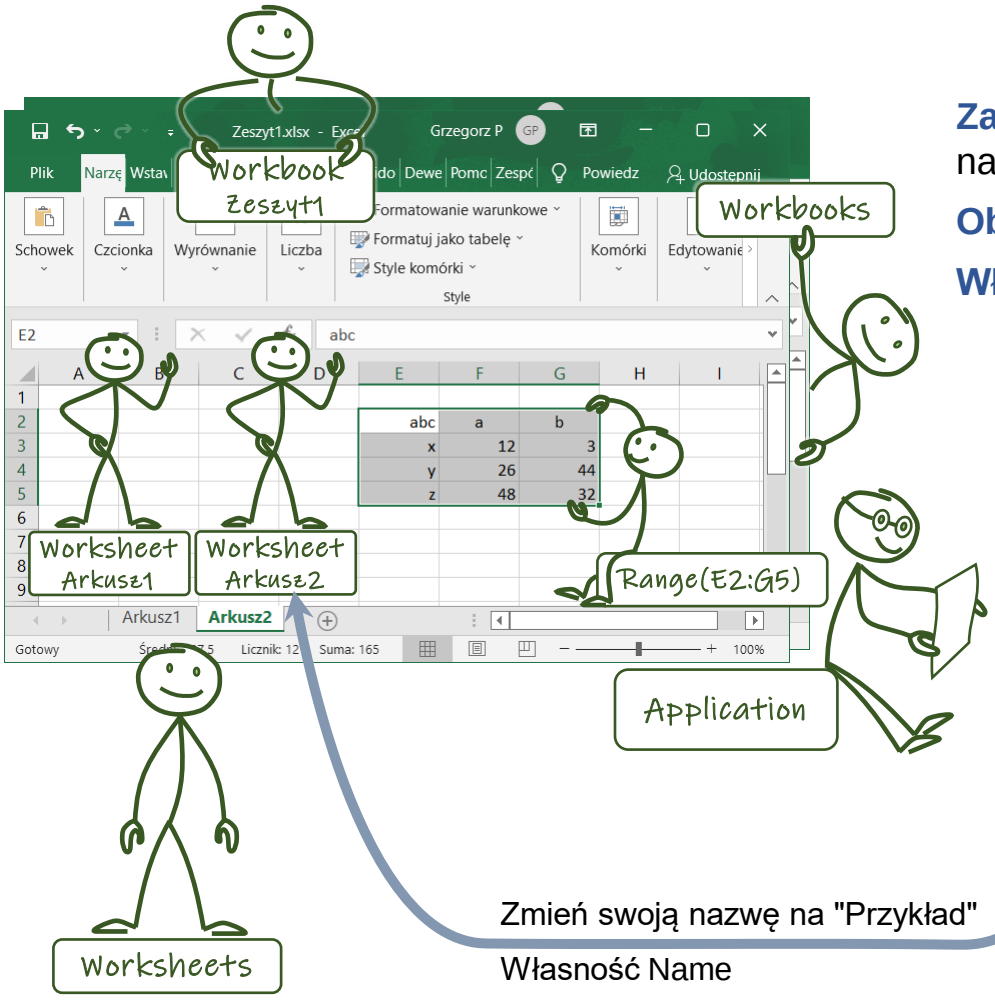
Wybrane metody

- **Activate** – aktywuje zakres (należy używać do wyboru komórki arkusza),
- **Clear** – czyści zakres (wartości, formuły, formaty),
- **ClearContents** – czyści zawartość zakresu (tylko wartości i formuły),
- **Copy(***zakres***), Cut(***zakres***), PasteSpecial** – kopiuje/wycina/wkleja zakres do innego *zakresu* lub schowka, jeżeli *zakres* docelowy nie został określony,
- **Delete(***przesunięcie***)** – usuwa zakres, pozostałe komórki są przenoszone w kierunku określonym przez *przesunięcie* (xlShiftToLeft, xlShiftUp),
- **FunctionWizard** – wyświetla okienko asystenta wstawiania funkcji,
- **Insert(***przesunięcie***)** – wstawia zakres, pozostałe komórki są przenoszone w kierunku określonym przez *przesunięcie* (xlShiftToRight, xlShiftDown),
- **Merge** – łączy komórki zakresu,
- **Select** – wybiera (zaznacza) zakres,
- **Speak** – odczytuje na głos zawartość komórek w zakresie.

Object Browser – narzędzie dostępne w edytorze VBA, które pozwala na przeglądanie dostępnych klas, ich własności i metod.



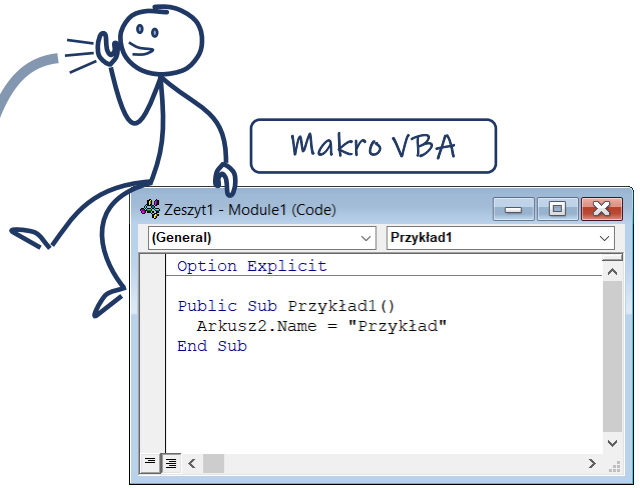
Idea pracy z obiektami – przykład I



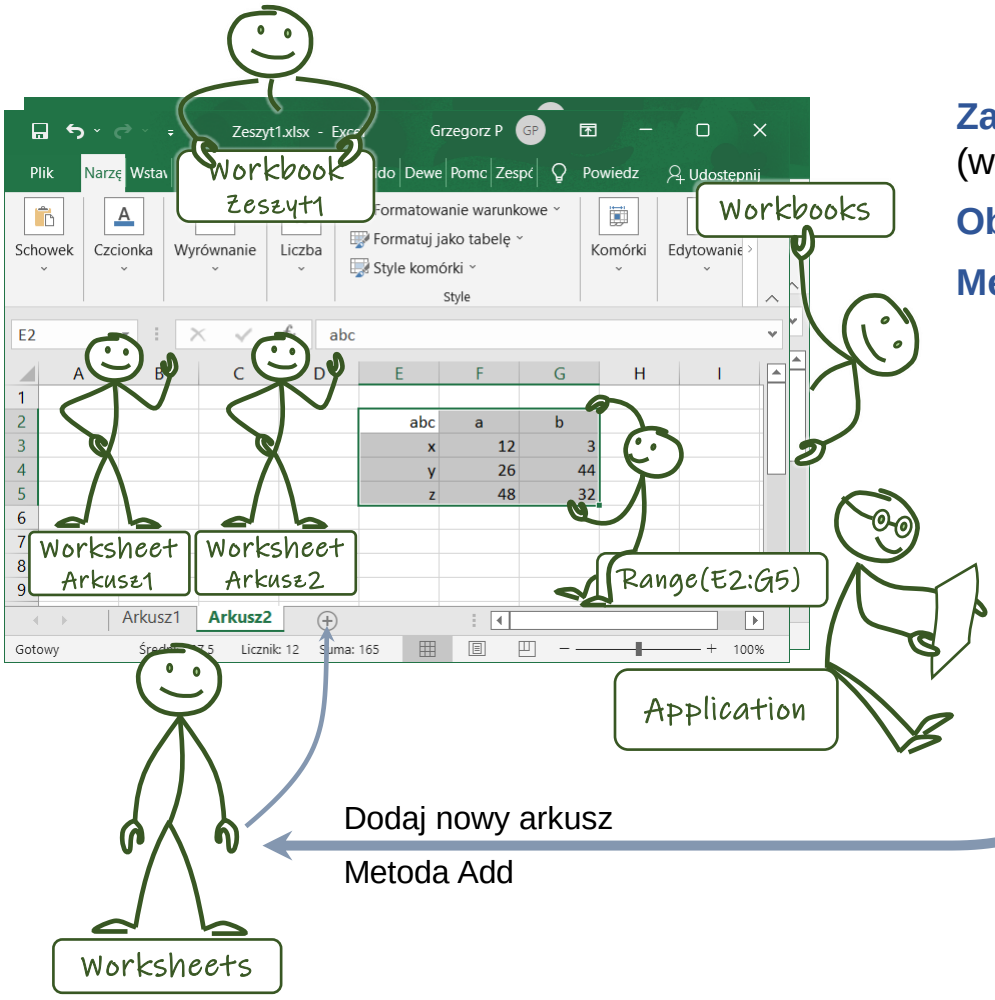
Zadanie: Zmienić nazwę arkusza "Arkusz2" na "Przykład".

Obiekt: Worksheet Arkusz2.

Własność: Name.



Idea pracy z obiektami – przykład II

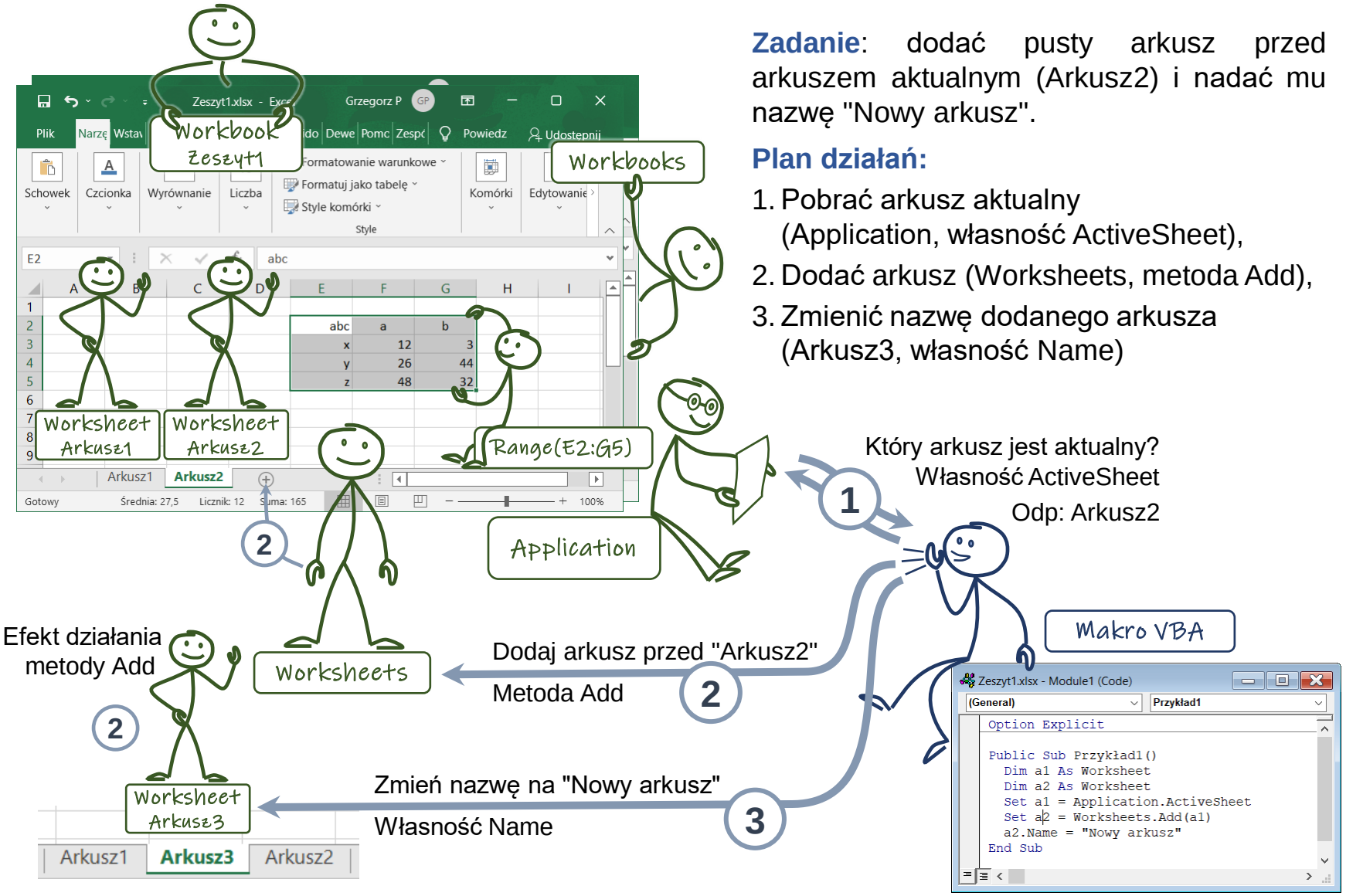


Zadanie: Dodać nowy arkusz do dokumentu (workbook'a) "Zeszyt1".

Obiekt: Worksheets z "Zeszyt1"

Metoda: Add

Idea pracy z obiektami – przykład III



Zadanie: dodać pusty arkusz przed arkuszem aktualnym (Arkusz2) i nadać mu nazwę "Nowy arkusz".

Plan działań:

1. Pobrać arkusz aktualny (Application, własność ActiveSheet),
2. Dodać arkusz (Worksheets, metoda Add),
3. Zmienić nazwę dodanego arkusza (Arkusz3, własność Name)

Podstawowe zasady programowania w VBA

- Wielkość liter nie jest rozróżniana (dotyczy wszystkich elementów programu: nazw instrukcji, elementów predefiniowanych i definiowanych przez użytkownika),
- Nazwy definiowanych elementów muszą zaczynać się od litery, nie mogą zawierać znaków specjalnych (spacja , * ; . = itp.),
- Pojedyncza instrukcja znajduje się w jednym wierszu programu,
- W celu podziału instrukcji na kilka wierszy należy zastosować symbol kontynuacji „ _” (spacja i znak podkreślenia),
- W jednym wierszu można umieścić kilka instrukcji rozdzielając je znakiem „:” (zmniejsza czytelność kodu, należy stosować tylko w szczególnych przypadkach),
- Puste wiersze są pomijane, mogą być wykorzystane do podziału kodu na mniejsze bloki (istotne z punktu widzenia czytelności programu),
- Słowa kluczowe języka (nazwy instrukcji, elementy nagłówków, itp.) są zastrzeżone i nie mogą być wykorzystane jako nazwy elementów definiowanych w programie,
- Tekst rozpoczynający się znakiem apostrofu jest komentarzem, nie wpływa na sposób wykonania programu.

Notacja

- **`Słowa kluczowe języka`** (np. nazwy instrukcji) – czcionka prosta, pogrubienie,
- `Elementy predefiniowane` (np. nazwy standardowych funkcji) – czcionka prosta, bez pogrubienia,
- `[elementy opcjonalne]` – nawiasy kwadratowe,
- Elementy alternatywne (należy wybrać jeden z nich) są rozdzielane pionową kreską (znak |),
- *`<nazwa>`* – tekst w nawiasach ostrych, kursywa,
- *`opis`* – kursywa.

Przykład

```
[Private|Public] Sub <nazwa>
```

- Pierwsze słowo to **`Private`** lub **`Public`**, jest opcjonalne (nawiasy kwadratowe)
- Słowo **`Sub`** musi wystąpić zawsze
- Po słowie **`Sub`** występuje nazwa wprowadzana przez użytkownika
- **`Private`**, **`Public`** i **`Sub`** to słowa kluczowe (pogrubienie)

Odwołania do obiektów

Odwołanie do własności i metody obiektu

```
nazwa_obiektu.nazwa_własności
nazwa_obiektu.nazwa_metody
```

Przykłady

`Application.ActiveSheet` – odwołanie do aktywnego arkusza

`Application.ActiveSheet.Name` – odwołanie do nazwy aktywnego arkusza

`Application.Worksheets.Add` – wywołanie metody „Add” (nowy arkusz)

Odwołanie do kolekcji

```
kolekcja(indeks)
```

`indeks` określa element, jego postać zależy od rodzaju kolekcji.

Przykłady

`Application.Workbooks(1)` – odwołanie do pierwszego dokumentu

`Application.Workbooks(”Z1.xlsm”)` – odwołanie do dokumentu „Z1.xlsm”

`Application.Workbooks(”Z2.xlsm”).Worksheets(”Arkusz1”)` – odwołanie do arkusza „Arkusz1” w dokumencie „Z2.xlsm”

Obiekty domyślne

Obiekty domyślne mogą być pomijane przy odwołaniach do ich własności obiektowych.

Obiekty domyślne w VBA

- Obiekt **Application** zawsze jest obiektem domyślnym i może być pominięty w większości odwołań, stąd odwołania postaci:

```
Application.Selection
Application.Workbooks(”Zeszyt1.xlsm”)
```

  mogą być skrócone do:

```
Selection
Workbooks(”Zeszyt1.xlsm”)
```

- Obiekty **Workbook** i **Worksheet** domyślnie odwołują się do aktywnego dokumentu oraz arkusza, stąd odwołanie postaci:

```
ActiveWorkbook.Worksheets(”Arkusz2”)
ActiveSheet.Range(”A5:C7”)
```

  mogą być skrócone do:

```
Worksheets(”Arkusz2”)
Range(”A5:C7”)
```

Typy danych

Typ danych określa zakres wartości, które może przyjmować dany element programu (np. własność obiektu). Każdy element przechowujący dane ma określony typ.

Predefiniowane typy danych VBA

- **`Byte`** – bajt, liczba całkowita `0 .. 255`
- **`Integer`** – liczba całkowita `-32 768 .. 32 767`
- **`Long`** – długa liczba całkowita `-2 147 483 648 .. 2 147 483 647`
- **`Single`** – liczba rzeczywista (pojedyncza precyzja , 8 cyfr) `1.40e-45 .. 3.40e38`
- **`Double`** – liczba rzeczywista (podwójna precyzja, 16 cyfr) `4.94e-324 .. 1.79e308`
- **`Currency`** – waluta (4 miejsca po przecinku)
  `-922 337 203 685 477.5808 .. 922 337 203 685 477.5807`
- **`Boolean`** – logiczny, dozwolone dwie wartości: `True` i `False`
- **`Date`** – data/czas `1 stycznia 100 roku .. 31 grudnia 9999`
- **`String`** – ciąg znaków, wartości podawane w cudzysłowach, maksymalnie 2 bln.
- **`Object`** – wskaźnik na dowolny obiekt
- **`Variant`** – dowolny typ, może przechowywać każdą z powyższych wartości

Uwaga: separatorem części ułamkowej w kodzie programu jest kropka

`1.756e02=1.756·10²=175.6, 2.5e-03=2.5·10⁻³=0.0025`

Typy danych – przykłady

`127` – liczba całkowita (**`Byte`**, **`Integer`** lub **`Long`**)

`5786` – liczba całkowita (**`Integer`** lub **`Long`**)

`"127"` – łańcuch znakowy (**`String`**)

`12.576` – liczba rzeczywista (**`Single`** lub **`Double`**, ewentualnie **`Currncy`**)

`True` – wartość logiczna (**`Boolean`**)

`"False"` – łańcuch znakowy (**`String`**)

`253,8` – nieprawidłowa wartość (przecinek zamiast kropki)

`25.39e-11` – liczba rzeczywista (**`Single`** lub **`Double`**, ewentualnie **`Currnecy`**)

`#20/5/2019 8:20:00 PM#` – data i czas (**`Date`**)

`ActiveCell` – własność Application, obiekt (**`Object`**, dokładnie Range)

`Selection` – własność Application, obiekt (**`Object`**, dokładnie Range)

`Worksheets(1)` – własność Application, obiekt (**`Object`**, dokładnie Worksheet)

`ActiveCell.Value` – zależne od wartości w komórce, **`Variant`**

Instrukcja przypisania, zmiana własności

Instrukcja przypisania

```
nazwa_elementu = wartość
```

Instrukcja przypisania nadaje pewnemu elementowi (np. własności obiektu) określoną wartość. Typ przypisywanej wartości musi być zgodny z typem elementu.

Przykład 1 – ustawienie wartości w aktywnej komórce arkusza

```
ActiveCell.Value = 125
ActiveCell.Value = ”Suma”
```

Przykład 2 – zmiana czcionki w komórce "C7"

```
Range(”C7”).Font.Color = RGB(255, 0, 0)
Range(”C7”).Font.Color = vbRed
Range(”C7”).Font.Name  = ”Courier New”
```

Uwaga 1.: Font jest własnością obiektową, zawiera własności Color, Name, Size, itp.

Uwaga 2.: Kolor można określić używając funkcji `RGB`, podając jako parametry nasycenie składowej czerwonej, zielonej i niebieskiej lub posługując się zdefiniowanymi stałymi o nazwach zgodnych ze schematem `vb<nazwa_koloru>`.

```
[Private|Public] Sub <nazwa>([arg1, arg2, … , argN])
  miejsce na kod procedury
End Sub
```

- **Sub** jest słowem kluczowym języka określa nagłówek procedury (subroutine)
- **Private** oznacza procedurę prywatną, dostępną tylko w module
- **Public** oznacza procedurę publiczną, dostępną w dowolnym miejscu programu
- Słowa **Private** i **Public** są opcjonalne, domyślnie procedury są publiczne
- *<nazwa>* jest nazwą nadawaną przez użytkownika (patrz punkt 2., s.23)
- *arg1, arg2, … argN* to opcjonalna lista argumentów zdefiniowanych jako:

  `[Optional] <nazwa_argumentu> As typ [= wartość_domyślna]`

- **Optional** oznacza argument opcjonalny, który może być pominięty podczas wywołania (w takim przypadku przyjmuje *wartość_domyślną*
- Procedura wykonuje kolejne instrukcje do wystąpienia **End Sub**
- Działanie procedury można przerwać umieszczając w kodzie instrukcję **Exit Sub**

Wywołanie procedury

Schemat wywołania procedury

```
<nazwa_procedury> [arg1, arg2, … , argN]
```

- W wywołaniu procedury nie występują nawiasy
- `<nazwa_procedury>` określa procedurę standardową lub zdefiniowaną przez użytkownika i dostępną w aktywnym arkuszu
- `arg1, … argN` odpowiadają kolejnym argumentom zgodnie z definicją zapisaną w nagłówku procedury
- Każdy argument opcjonalny może być pominięty (puste miejsce)
- Argumenty procedury można podać w dowolnej kolejności określając ich nazwy zgodnie ze schematem:

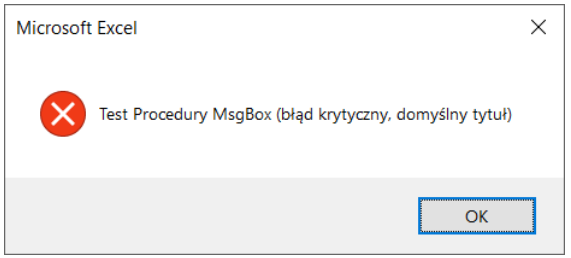
```
<nazwa_argumentu> := wartość
```

Przykład – MsgBox

Nagłówek

```
MsgBox(prompt As String, _
       Optional buttons As VbMsgBoxStyle = vbOKOnly, _
       Optional title As String = "Microsoft Excel")
```

- `prompt` – tekst wyświetlany w oknie komunikatu
- `buttons` – zestaw przycisków dostępnych w oknie (`vbOKOnly` – tylko przycisk OK) oraz typ okna (wyświetlana ikona). Dostępne wartości: `vbCritical`, `vbQuestion`, `vbExclamation`, `vbInformation` (błąd krytyczny, pytanie, ostrzeżenie, informacja)
- `title` – tytuł okna (domyślnie „Microsoft Excel”)



Przykład wywołania

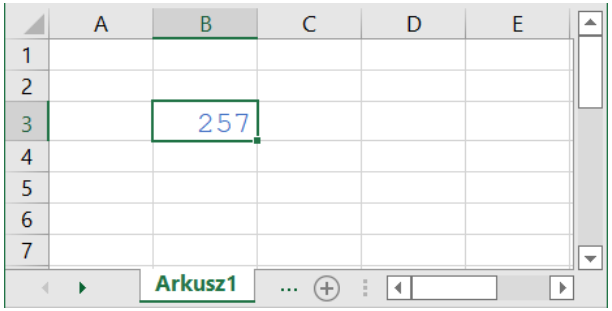
```
MsgBox "Test Procedury MsgBox"
MsgBox "Test Procedury MsgBox", vbCritical
MsgBox "Test Procedury MsgBox", vbQuestion, "Mój komunikat"
MsgBox "Test Procedury MsgBox", , "Mój komunikat"
MsgBox buttons:=vbExclamation, title:="Mój komunikat", _
       prompt:="Test Procedury MsgBox"
```

Kod dostępny na stronie przedmiotu

Przykłady makr

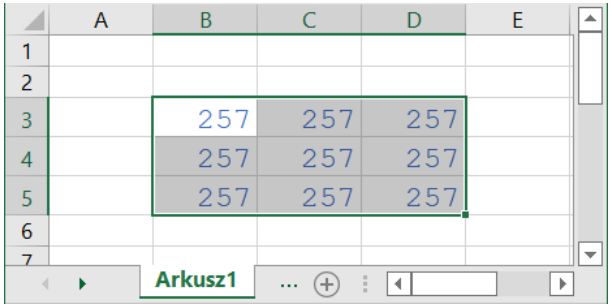
Modyfikacja aktywnej komórki: ustawienie wartości 257, zmiana czcionki na Courier New, niebieska, rozmiar 14pt.

```
Public Sub Przykład1()
  ActiveCell.Value = 257
  ActiveCell.Font.Name = "Courier New"
  ActiveCell.Font.Color = vbBlue
  ActiveCell.Font.Size = 14
End Sub
```



Modyfikacja wszystkich komórek w aktualnym zakresie.

```
Public Sub Przykład2()
  Selection.Value = 257
  Selection.Font.Name = "Courier New"
  Selection.Font.Color = vbBlue
  Selection.Font.Size = 14
End Sub
```



Kody dostępne na stronie przedmiotu

Przykłady makr

Przepisanie zawartości zakresu `A1:A5` z `Akrusz1` do zakresu `C3:E7` na `Arkusz2`, aktywacja `Arkusza2`.

```
Public Sub Przykład3()
  Worksheets("Arkusz2").Range("C3:E7").Value = _
  Worksheets("Arkusz1").Range("A1:A5").Value
  Worksheets("Arkusz2").Activate
End Sub
```

Uwaga: pojedyncza instrukcja zapisana w kilku wierszach, użyty symbolu kontynuacji: " _".

Przepisanie zawartości aktualnego zakresu o dwie kolumny w prawo.

```
Public Sub Przykład4()
  Selection.Offset(0, 2).Value = Selection.Value
End Sub
```

Nazwa użytkownika, na którego został zarejestrowany program.

```
Public Sub Przykład5()
  MsgBox Application.UserName, vbInformation, "Nazwa użytkownika"
End Sub
```

Uwaga: przy odwołaniu do własności UserName nie można pominąć obiektu Application.

Kody dostępne na stronie przedmiotu

Instrukcja `With`

```
With <obiekt>
  odwołania do własności i metod
End With
```

Wewnątrz instrukcji **`With`** wszystkie odwołania do własności i metod obiektu mogą być skrócone do postaci:

```
.nazwa_własności
.nazwa_metody
```

Przykład

Procedury ustawiają wartość aktywnej komórki na 257, zmieniają kolor i rozmiar czcionki.

```
Public Sub Przykład1a()
  ActiveCell.Value = 257
  ActiveCell.Font.Color = vbBlue
  ActiveCell.Font.Size = 14
End Sub
```

```
Public Sub Przykład1b()
  With ActiveCell
    .Value = 257
    .Font.Color = vbBlue
    .Font.Size = 14
  End With
End Sub
```

Instrukcja przypisania, operatory arytmetyczne

Instrukcja przypisania

Instrukcja przypisania pozwala na modyfikację każdego elementu programu, który nie został zdefiniowany z atrybutem read-only (tylko do odczytu)

```
<element> = wyrażenie
```

Uwaga: jako pierwsze zawsze wykonywane jest wyrażenie po prawej stronie instrukcji przypisania, do elementu po stronie lewej przypisywany jest wynik wyrażenia.

Operatory arytmetyczne

`+`	dodawanie	`\`	dzielenie całkowite
`-`	odejmowanie	`mod`	dzielenie modulo
`*`	mnożenie	`^`	potęgowanie
`/`	dzielenie	`( )`	grupowanie działań

Operacje wykonywane są od lewej do prawej z zachowaniem priorytetów (po pierwsze potęgowanie, następnie mnożenie i dzielenie, później dodawane i odejmowanie). Nie można pomijać symboli działań, nawiasy okrągłe zmieniają kolejność operacji.

Uwaga: $x + 5/3*y = x + \frac{5}{3}y$, $(x + 5)/(3*y) = \frac{x+5}{3y}$

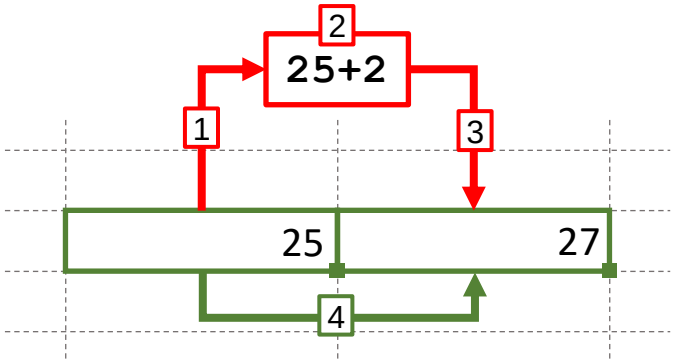
Przykład

Procedura pobiera wartość aktualnej komórki, powiększa ją o dwa, wynik zapisuje w komórce znajdującej się na prawo (ten sam wiersz, następna kolumna) i zaznacza ją.

Odwołania

- Aktualnie wybrana komórka: `ActiveCell`
- Odwołanie do wartości aktualnej komórki: `ActiveCell.Value`
- Odwołanie do wartości komórki sąsiedniej: `ActiveCell.Offset(0,1).Value`
- Zaznaczenie komórki sąsiedniej: `ActiveCell.Offset(0,1).Activate`

```
Public Sub InkrementujWPrawo()
  With ActiveCell
3 .Offset(0, 1).Value = .Value + 2
4 .Offset(0, 1).Activate
  End With
End Sub
```



Kod dostępny na stronie przedmiotu

Zmienna – element języka programowania używany do przechowywania danych. Podczas pracy programu wartości zmiennej mogą ulegać modyfikacji.

Stała – symbol reprezentujący pewną niezmienną wartość, która nie może być zmodyfikowana podczas pracy programu.

Schemat deklaracji zmiennej

```
Dim <nazwa> As typ
```

Schemat deklaracji stałej

```
Const <nazwa> [As typ] = wartość
```

<nazwa> musi być unikalną (nie mogą istnieć dwa elementy o takiej samej nazwie), powinna spełniać warunki opisane w punkcie 2 na s.23.

Przykłady

```
Dim x As Integer
Dim komórka As Range
Const VAT As Single = 0.23
```

Uwaga: Deklaracja zmiennych nieobiektowych nie jest wymagana. Niezadeklarowana zmienna otrzymuje typ **`Variant`**. Deklaracja zmiennych może być wymuszona przez umieszczenie dyrektywy **`Option Explicit`** na początku modułu (**`Tools -> Options -> Require variable declaration`**).

Ustawianie wartości zmiennych

Ustawienie wartości zmiennej nieobiektowej

```
<nazwa> = wyrażenie
```

Ustawienie wartości zmiennej obiektowej

```
Set <nazwa> = wyrażenie
```

<nazwa> jest nazwą zmiennej

wyrażenie jest dowolnym wyrażeniem VBA (w tym również wartością stałą, nazwą zmiennej, itp.) o typie zgodnym z typem zmiennej.

Przykłady

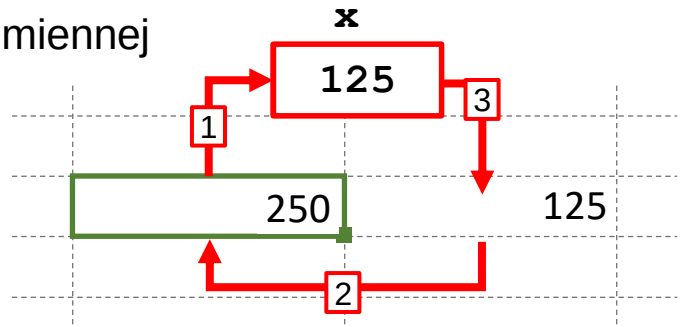
```
Dim x As Integer
Dim y As Integer
Dim k As Range

x = 5
y = 29 + x ^ 2
Set k = Worksheets("Arkusz5").Range("C3:E5")
```

Przykład

Procedura zamienia miejscami wartość komórki aktywnej i komórki znajdującej się na prawo od niej (ten sam wiersz kolejna kolumna).

Algorytm

1. Odczytaj wartość komórki aktywnej do zmiennej
2. Zapisz w komórce aktywnej wartość komórki po prawej
3. Zapisz w komórce po prawej wartość zapisaną w zmiennej




```
Public Sub ZamieńWPrawo1()
  Dim x As Integer
  x = ActiveCell.Value
  ActiveCell.Value = ActiveCell.Offset(0, 1).Value
  ActiveCell.Offset(0, 1).Value = x
End Sub
```

Kod dostępny na stronie przedmiotu

Uwaga: Procedura działa prawidłowo dla wartości typu **`Integer`** (dla innych wartości zgłasza błąd).

Do przemyślenia (1): Zamiana dowolnych wartości.

Do przemyślenia (2): Zamiana całego zakresu.

Funkcje VBA

```
[Private|Public] Function <nazwa>([arg1,… , argN]) As typ
  miejsce na kod funkcji
  <nazwa> = wartość
End Function
```

- **Function** jest słowem kluczowym języka określa nagłówek funkcji
- **Private|Public** są opcjonalne, określają dostępność funkcji, funkcja publiczna jest traktowana jak standardowa funkcja arkuszowa i może być użyta w formułach
- *<nazwa>* jest nazwą nadawaną przez użytkownika (patrz punkt 2., s.23)
- *typ* określa typ wartości zwracanej jako wynik działania funkcji
- Wynik funkcji jest określany przez przypisanie *wartości* do *<nazwy>*
- *arg1,… argN* to opcjonalna lista argumentów, definicja jak w procedurze (s.30)
- Funkcja wykonuje kolejne instrukcje do wystąpienia **End Function**
- Działanie funkcji można przerwać umieszczając w kodzie instrukcję **Exit Function**
- Obsługa błędów może być realizowana tak jak w przypadku procedur

Wywołanie funkcji

Schemat wywołania funkcji

```
zmienna = <nazwa_funkcji>[(arg1, … , argN)]
```

- W wywołaniu funkcji nawiasy są wymagane jeżeli następuje przekazanie argumentów. W przypadku funkcji bezargumentowej są dozwolone, ale niewymagane.
- *<nazwa_funkcji>* określa procedurę standardową lub zdefiniowaną przez użytkownika i dostępną w aktywnym arkuszu.
- *zmienna* oznacza nazwę zmiennej, w której zostanie umieszczona wartość zwracana przez funkcję.
- *arg1, … argN* odpowiadają kolejnym argumentom zgodnie z definicją zapisaną w nagłówku procedury.
- Każdy argument opcjonalny może być pominięty (puste miejsce)
- Argumenty funkcji można podać w dowolnej kolejności określając ich nazwy zgodnie ze schematem:

```
<nazwa_argumentu> := wartość
```

Komunikacja z użytkownikiem – InputBox

Funkcja `InputBox`

```
InputBox(prompt As String,
         Optional title As String = "Microsoft Excel",
         Optional default As String = "") As String
```

Metoda `InputBox` (klasa Application)

```
InputBox(prompt As String,
         Optional title As Variant = "Microsoft Excel",
         Optional default As Variant = "", …
         Optional type As Variant) As Variant
```

- `prompt` – tekst wyświetlany w oknie
- `title` – tytuł okna (domyślnie „Microsoft Excel”)
- `default` – wartość domyślna
- `type` – typ odczytywanej wartości (tylko metoda `InputBox`): 0 – formuła, 1 – liczba, 2 – tekst, 4 – wartość logiczna, 8 – zakres (Range).

Uwaga: Zarówno funkcja jak i metoda `InputBox` ma cztery dodatkowe parametry (położenie okna i odwołanie do systemu pomocy), które nie zostały opisane.

Przykład – InputBox

Procedura wypełnia wskazany zakres określoną wartością. Do odczytu zakresu wykorzystano metodę `InputBox`, do odczytu wartości funkcję `InputBox`.

```
Public Sub Wypełnij()
  Dim Zakres As Range
  Dim wartość As String
  Set Zakres = Application.InputBox("Wskaż zakres", type:=8)
  wartość = InputBox("Podaj wartość")
  Zakres.Value = wartość
End Sub
```

Uwaga: Procedura zgłasza błąd w przypadku naciśnięcia przycisku Anuluj w oknie InputBox (metoda aplikacji). Sposób na wyeliminowanie tego problemu zostanie przedstawiony na wykładzie drugim.

Kod dostępny na stronie przedmiotu

Funkcje użytkownika – przykład

```
Const DomyślnyVAT As Single = 0.23

Public Function CenaBrutto1(netto As Currency,
                            vat As Single) As Currency
  CenaBrutto1 = netto + netto * vat
End Function

Public Function CenaBrutto2(netto As Currency,
                            Optional vat As Single = DomyślnyVAT) As Currency
  CenaBrutto2 = netto + netto * vat
End Function
```

	A	B	C	D	E	F	G	H
1	**Funkcja CenaBrutto1**					**Funkcja CenaBrutto2**		
2	**Netto**	**VAT**	**Brutto**			**Netto**	**Brutto**	
3	127,00 zł	23%	156,21 zł			127,00 zł	156,21 zł	
4	250,00 zł	23%	307,50 zł			250,00 zł	307,50 zł	
5	35,00 zł	8%	37,80 zł			35,00 zł	37,80 zł	
6	875,00 zł	23%	1 076,25 zł			875,00 zł	1 076,25 zł	
7	1 230,00 zł	8%	1 328,40 zł			1 230,00 zł	1 328,40 zł	
8	27,00 zł	23%	33,21 zł			27,00 zł	33,21 zł	
9								

=CenaBrutto2(F3)

=CenaBrutto2(F5;0,08)

=CenaBrutto1(A3;B3)

Kod dostępny na stronie przedmiotu