Java - podstawy

Marcin Szupke

Java

Definicja (wikipedia)

Java - współbieżny, oparty na klasach, obiektowy język programowania ogólnego zastosowania. Został stworzony przez grupę roboczą pod kierunkiem Jamesa Goslinga z firmy Sun Microsystems. Java jest językiem tworzenia programów źródłowych kompilowanych do kodu bajtowego, czyli postaci wykonywanej przez maszynę wirtualną. Język cechuje się silnym typowaniem. Jego podstawowe koncepcje zostały przejęte z języka Smalltalk (maszyna wirtualna, zarządzanie pamięcią) oraz języka C++ (duża część składni i słów kluczowych).

Historia

- Java 1.02 (1996) - fajna nazwa i logo. Jej używanie dawało dużo radości. Masa błędów. Zdecydowanie najciekawsze są obiekty

- Java 1.1 (1997) - więcej możliwości, nieco bardziej przyjazna dla programistów. Staje się bardzo popularna. Lepszy kod do obsługi graficznego interfejsu użytkownika

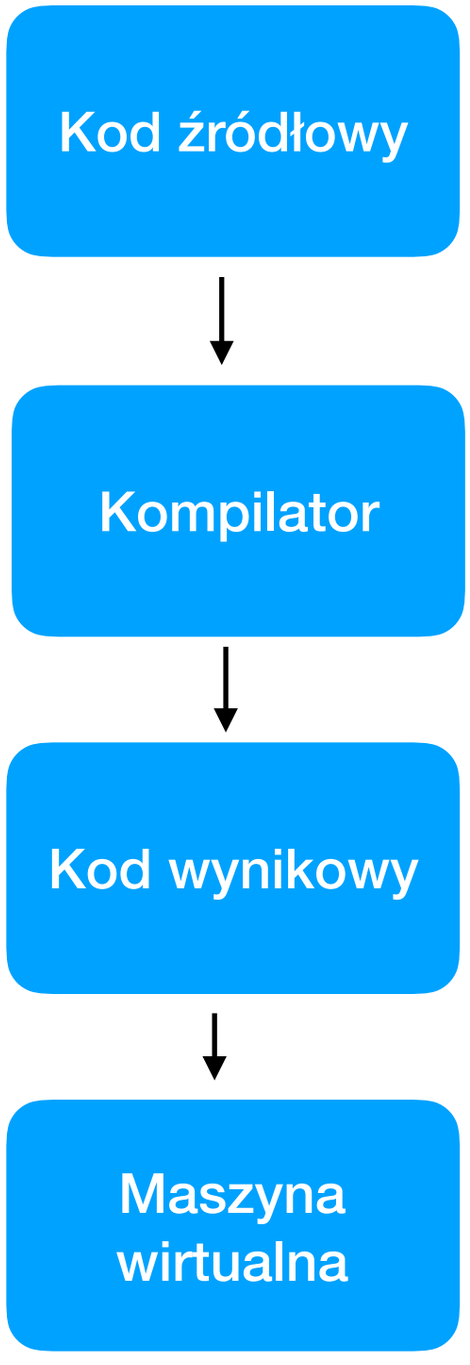
- Java 2 (wersje 1.2 - 1.4, 1998 - 2002) - poważny język programowania o olbrzymich możliwościach. Dostępna w trzech wersjach: Micro Edition (J2ME), Standard Edition (J2SE) oraz Enterprise Edition (J2EE). Stała się preferowanym językiem dla aplikacji biznesowych oraz aplikacji dla urządzeń przenośnych

Historia

- Java 5.0 (wersja 1.5 i kolejne, 2004) - ma jeszcze większe możliwości i w większym stopniu ułatwia pisanie aplikacji. W Javie 5.0 oprócz dodania ponad tysiąca nowych klas wprowadzono także zmiany w samym języku, które teoretycznie miały ułatwiać życie programistom i rozszerzyć go o nowe możliwości znane z innych, popularnych języków programowania

- Java 5.0 doczekała się 85 aktualizacji, z których ostatnia została wydana w kwietniu 2015 roku

Jak działa Java?



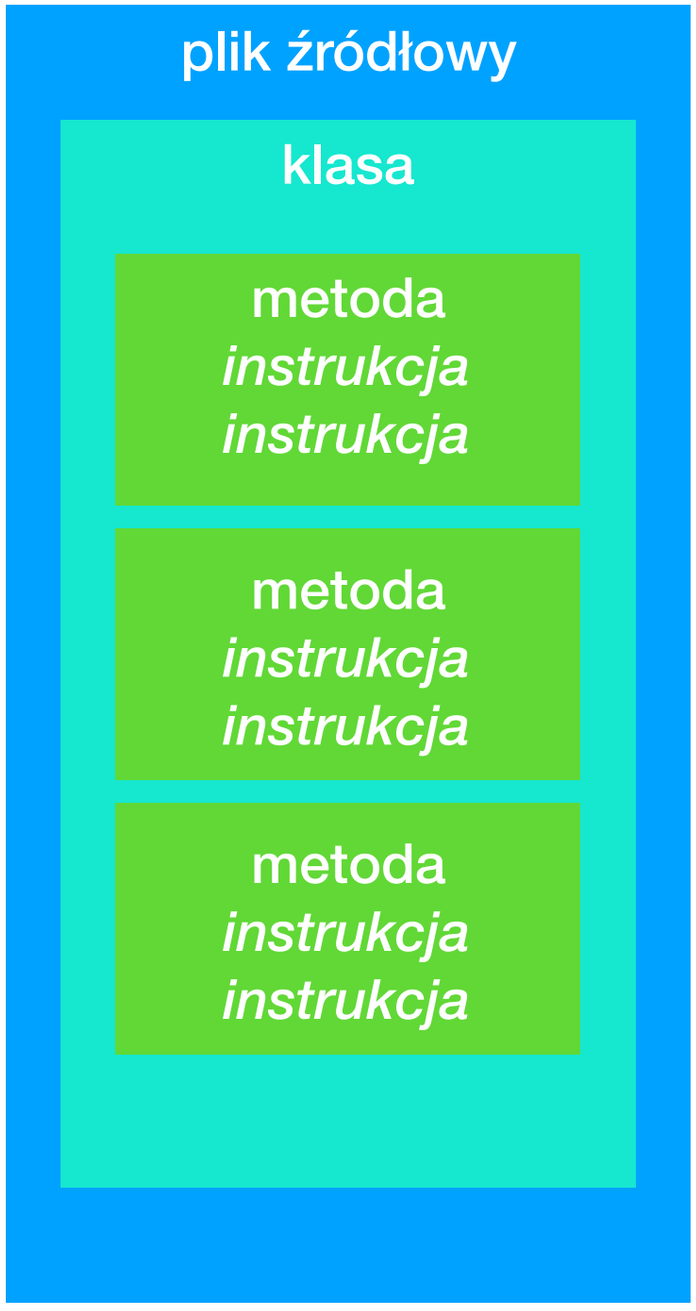
Stworzenie dokumentu źródłowego w języku Java

Przetwarza dokument i sprawdza, czy w programie nie ma błędów i nie dopuści do jego skompilowania aż do momentu, gdy uzyska pewność, że program uruchomi się poprawnie

Kompilator tworzy nowy dokument, którego zawartość stanowi skompilowany kod bajtowy Javy. Java będzie w stanie zinterpretować ten dokument do postaci, którą będzie w stanie wykonać. Skompilowany kod bajtowy jest niezależny od platformy systemowej

Urządzenia (komputery, smartfony, etc.) nie mają fizycznej maszyny Javy, dysponują jednak maszynami wirtualnymi (zaimplementowanymi w formie programów). Te wirtualne maszyny odczytują i wykonują kody bajtowe

Struktura kodu



Plik źródłowy zawiera definicję klasy

Klasa reprezentuje pewien element programu (lub jego całość).

Wewnątrz klasy umieszcza się jedną lub kilka metod.

Metody składają się z instrukcji, które określają sposób działania metody

Anatomia klasy

```java
public class Main {

    public static void main(String[] args) {

        System.out.println("Hello, world!");

    }
}
```

Instrukcje

Deklaracje, przypisania, wysłania metod…

```
int x;
String name = "Janek";

x = 4 + 11;
System.out.println("x = " + x);

double d = Math.random();
// komentarz
```

Pętle

```
int x = 11;
while (x > 0) {
    System.out.println("Aktualnie x = " + x);
    x = x - 1;
}
```

```
for (int i = 0; i < 11; i++) {
    System.out.println("Aktualnie i = " + i);
}

int y = 11;
do {
    System.out.println("Aktualnie y = " + y);
    y = y - 1;
} while (y > 11);
```

Instrukcje warunkowe

```java
int x = 1;
if (x > 0) {
    System.out.println("x jest większe niż 0");
}

if (x > 0) {
    System.out.println("x jest większe niż 0");
} else {
    System.out.println("x jest mniejsze lub równe 0");
}

if (x > 0) {
    System.out.println("x jest większe niż 0");
} else if (x < 0) {
    System.out.println("x jest mniejsze niż 0");
} else {
    System.out.println("x jest równe 0");
}
```

Składnia

- Każda instrukcja musi kończyć się średnikiem
- Komentarze jednowierszowe zaczynają się od dwóch znaków ukośnika
- Zmienne deklaruje się podając nazwę oraz typ
- Definicje klas i metod należy umieszczać wewnątrz nawiasów klamrowych

Testy logiczne

- Testy logiczne przeprowadza się do sprawdzenia wartości zmiennej przy wykorzystaniu operatora porównania, takiego jak:
  - < (mniejszy niż)
  - <= (mniejszy, równy)
  - > (większy niż)
  - >= (większy, równy)
  - == (równy)
  - != (różny)

Zmienne

- Zmienna służy do przechowywania danych

- Java jest językiem opartym na silnym typowaniu, w którym każde wyrażenie ma ustalony typ i nie można go używać w kontekście przeznaczonym dla innych typów

```
int dataValue;
dataValue = 100;
```

- Zmienne mogą być modyfikowane

Nazwy zmiennych

- Nazwy zmiennych bazują na kombinacji zasad i konwencji
  - Zasady zezwalają na używanie liter, cyfr, $ i _
    - Według konwencji używa się tylko liter i cyfr
  - Zasady zabraniają by nazwy rozpoczynały się od cyfry
    - Według konwencji nazwy zaczynają się od litery
- Według konwencji nazwy zmiennych pisane są w systemie notacji „camel case”
  - Pierwszy znak jest małą literą
  - Każde inne słowo, z którego składa się nazwa rozpoczyna się wielką literą
  - Wszystkie inne litery są małe

```
int bankAccountBalance;
```

Używanie zmiennych

```java
public class Main {

    public static void main(String[] args) {

        int mojaZmienna;

        mojaZmienna = 50;
        System.out.println(mojaZmienna);

        int innaZmienna = 100;
        System.out.println(innaZmienna);

        mojaZmienna = innaZmienna;
        System.out.println(mojaZmienna);

        innaZmienna = 200;
        System.out.println(innaZmienna);

        System.out.println(mojaZmienna);
    }
}
```

Typy podstawowe

Typ	Ilość bitów	Zakres wartości
Wartości logiczne i znaki		
boolean	zależy od JVM	true / false
char	16 bitów	0 do 65 535
Liczby całkowite		
byte	8 bitów	-128 do 127
short	16 bitów	-32 768 do 32 767
int	32 bity	-2 147 483 468 do 2 147 483 467
long	64 bity	-„bardzo dużo” do „bardzo dużo”
Liczby zmiennoprzecinkowe		
float	32 bity	różnie
double	64 bitów	różnie

Typy podstawowe - zapisywane jako wartość

Operatory arytmetyczne

- Podstawowe operatory
  - +, -, *, /, %
- Operatory prefix/postfix
  - ++, - -
- Złożone operatory przypisania
  - +=, -=, *=, /= %=

Podstawowe operatory matematyczne

	Operator	Przykład w świecie liczb zmiennoprzecinkowych	Przykład w świecie liczb całkowitych
Dodawanie	+	1.0 + 2.0 = 3.0	1 + 2 = 3
Odejmowanie	-	5.0 - 4.0 = 1.0	5 - 4 = 1
Mnożenie	*	4.0 * 2.0 = 8.0	4 * 2 = 8
Dzielenie	/	13.0 / 5.0 = 2.6	13 / 5 = 2
Modulo	%	13.0 % 5.0 = 3.0	13 % 5 = 3

Operator Prefix / Postfix

- ++ zwiększa wartość zmiennej o 1

- - - zmniejsza wartość zmiennej o 1

```
int myVal = 5;
System.out.println(++myVal);   6
System.out.println(myVal)      6
```

```
int myVal = 5;
System.out.println(myVal++);   5
System.out.println(myVal)      6
```

Złożone operatory przypisania

- Łączą ze sobą operacje matematyczne z przypisaniem
- Używają rezultatu prawej strony operacji do lewej strony
- Zapisują wynik do zmiennej po lewej stronie
- Dostępna dla wszystkich bazowych operatorów matematycznych:
  += -= *= /= %=

```
int myVal = 50;
myVal -= 5;
System.out.println(myVal)
```

45

```
int result = 100;
Int val1 = 5;
Int val2 = 10;
result /= val1 * val2;

System.out.println(result)
```

100 / 50

2

Konwersja typów

Niejawna konwersja typów - wykonywana automatycznie przez kompilator

```
int iVal = 50;
long lVal = iVal;
```

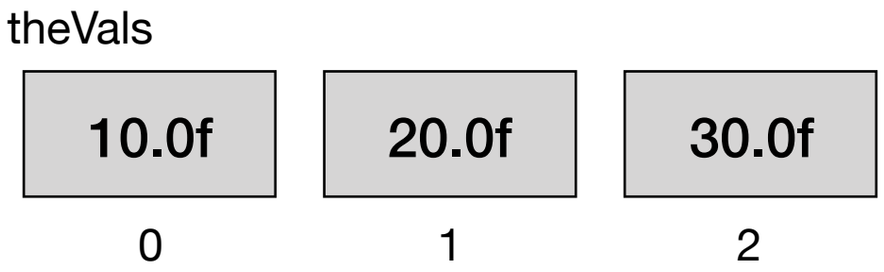
Jawna konwersja typów - wykonywana jawnie przez operator rzutowania

```
long lVal = 50;
int iVal = (int) lVal;
```

Tablice

Tablice to uporządkowana kolekcja elementów

- Każdy element jest dostępny przez indeks
- Indeksy numerowane są od 0 do ilości elementów minus 1
- Ilość elementów jest zapisana w wartości *length* tablicy




```
float[] theVals = new float[3];
theVals[0] = 10.0f;
theVals[1] = 20.0f;
theVals[2] = 30.0f;

float sum = 0.0f;
for (int i = 0; i < theVals.length; i++) {
    sum += theVals[i];
}

System.out.println(sum);
```

Pętla for-each

Wykonuje instrukcje dla każdego elementu tablicy

- Pobiera długość tablicy
- Obsługuje dostęp do każdej wartości tablicy

```
float[] theVals = new float[3];
theVals[0] = 10.0f;
theVals[1] = 20.0f;
theVals[2] = 30.0f;

float sum = 0.0f;
for (float currentVal : theVals) {
    sum += currentVal;
}

System.out.println(sum);
```

Switch

Instrukcja wielokrotnego wyboru, dzięki której można warunkowo wykonać pewne fragmenty kodu

- Tylko typy int i char mogą zostać użyte do testu warunku
- Znaleziony wzorzec może wykonać więcej niż jedną instrukcję
- Do przerwania wykonania instrukcji służy słowo kluczowe *break*
- Słowo kluczowe *default* jest używane do obsługi nieznalezienia wzorca

```
int testVal = 10;
switch (testVal) {
    case 0:
        System.out.println("Testowana wartość to 0");
    case 1:
        System.out.println("Testowana wartość to 1");
    case 2:
        System.out.println("Testowana wartość to 2");
    case 10:
        System.out.println("Testowana wartość to 10");
    case 11:
        System.out.println("Testowana wartość to 11");
    default:
        System.out.println("Testowana wartość to 0");
}
```

Zadania

1. Napisz program, który dla podanej liczby (np. 12345) wypisze ją w odwrotnej kolejności (54321)

2. Napisz program, który dla zadanej tablicy (np. [1,2,3,4,5]) policzy sumę wszystkich elementów (15)

3. Napisz program, który dla zadanej tablicy (np. [1,2,3,4,5]) policzy średnią z wszystkich elementów (3)

4. Napisz program, który wyznaczy indeks elementu tablicy (np. indeks dla liczby 9 w tablicy [1,4,5,9,12,99] to 3)

5. Napisz program, który wykona kopię tablicy element po elemencie

6. Napisz program, który wyznaczy najmniejszą i największą wartość w tablicy

7. Napisz program, który wykorzystując instrukcję switch sprawdzi, czy podana liczba jest parzysta czy nieparzysta

Klasa

- Java jest językiem zorientowanym obiektowo
- Obiekty hermetyzują dane, operacje i semantykę wykonania
  - Ukrywają szczegóły zapisywania i manipulacji danymi
  - Oddzielają to „co” może zostać wykonane od tego „jak” to się wykonuje
- Klasy dostarczają strukturę do opisywania i tworzenia obiektów
- Klasa to typ referencyjny

Klasa

Klasa składa się ze stanu i kodu wykonywalnego

- Składowe
    - Zapisują stan obiektu
- Metody
    - Kod wykonywalny, który manipuluje stanem i wykonuje operacje
- Konstruktory
    - Kod wykonywalny uruchamiany podczas tworzenia obiektu w celu ustawienia początkowego stanu

```java
public class Pies {

    private String rasa;
    private String imie;
    private int wielkosc;

    public Pies() {
        rasa = "Labrador";
        imie = "Azor";
        wielkosc = 50;
    }

    public void szczekaj() {
        // implementacja umiejętności szczekania
    }

}
```

Hermetyzacja i modyfikatory dostępu

- Hermetyzacja polega na ukrywaniu pewnych danych składowych lub metod danej klasy

- Java wykorzystuje modyfikatory dostępu by osiągnąć hermetyzację

Modyfikator	Widoczność	Używane w klasach	Używane w składowych
Bez modyfikatora	Tylko wewnątrz pakietu	Tak	Tak
public	Wszędzie	Tak	Tak
private	Tylko wewnątrz klasy	Nie*	Tak

Nazwy klas

- Nazwy klas podlegają tym samym zasadom co nazwy zmiennych
- Konwencja nazewnicza klas jest podobna do konwencji nazewniczej zmiennych z kilkoma różnicami
    - Używaj tylko liter i cyfr
    - Pierwszy znak zawsze jest literą
    - Używaj stylu „Pascal case”
        - Każde słowo rozpoczyna się z wielkiej litery
        - Wszystkie inne znaki pisane są małą literą
    - Używaj prostych, opisowych rzeczowników
    - Unikaj skrótów, chyba że są one częściej używane niż pełna nazwa (np. URL - Uniform Resource Locator)

Metody - wstęp

Kod wykonywalny, który manipuluje stanem i wykonuje operacje, która składa się z:

- Nazwy
    - Podlegają tym samym regułom i konwencjom co zmienne
        - Powinna być czasownikiem lub opisem akcji
- Typu zwracanego
    - Używaj void, gdy metoda nic nie zwraca
- Listy parametrów
    - Może być pusta
- Ciało metody jest wewnątrz nawiasów klamrowych

```
public void szczekaj() {
    System.out.println("Hau, hau!");
}
```

Kończenie działania metody

- Metoda kończy swoje działanie gdy:
  - Zostanie osiągnięty koniec metody
  - Zostanie wywołane słowo kluczowe *return*
  - Wystąpi błąd
- Gdy podczas wywołania metody nie wystąpi błąd, sterowanie wraca do wywołującego metodę

```
public void szczekaj() {

    if (zmeczenie >= 10) {
        return;
    }

    System.out.println("Hau, hau!");
    zmeczenie++;

    return;
}
```

Zwracanie wartości przez metody

Metody zwracają pojedynczą wartość

- Typu prostego
- Referencję do obiektu
- Referencję do tablicy

```
public class Pies {

    private int zmeczenie;

    public boolean czyZmeczony() {

        if (zmeczenie >= 10) {
            return true;
        }
        else {
            return false;
        }
    }

}
```

Referencje: this i null

Java dostarcza specjalnych referencji,
które mają zdefiniowane znaczenie:

- *this* - referencja do obecnego obiektu
  - Używany do redukcji dwuznaczności
  - Umożliwia obiektowi przekazanie siebie samego jako parametr
- *null* - referencja do nieutworzonego obiektu
  - Może zostać przypisany do każdego zmiennej typu referencyjnego

```java
public class Pies {

    private String rasa;
    private String imie;
    private int wielkosc;

    public Pies(String rasa, String imie, int wielkosc) {
        this.rasa = rasa;
        this.imie = imie;
        this.wielkosc = wielkosc;
    }
}
```

Akcesory i mutatory

Używa się ich jako wzorzec kontroli składowymi klasy

- Akcesory (gettery, metody dostępowe) zwracają wartości składowych
    - Metody, które nazywa się zgodnie z konwencją: getNazwaSkladowej
- Mutattory (settery, metody zmieniające) ustawiają wartości składowych
    - Metody, które nazywa się zgodnie z konwencją: setNazwaSkladowej

```
public class Pies {

    private String rasa;

    public String getRasa() {
        return rasa;
    }

    public void setRasa(String rasa) {
        this.rasa = rasa;
    }
}
```

Zadania

- Napisz klasy
  - Trojkat - składowe: podstawa, wysokość
  - Kwadrat - składowa: dlugoscBoku
  - Prostokat - składowe: dlugoscPierwszegoBoku, dlugoscDrugiegoBoku
  - Kolo - składowa: promien
- W każdej klasie zaimplementuj metody obliczPole i obliczObwod zwracające typ double
- Napisz klasę Main i zaimplementuj metodę main, w której utworzysz instancje klas Trojkat, Kwadrat, Prostokat

Ustalenie początkowego stanu

- Kiedy tworzymy obiekt, oczekujemy by jego stan był użyteczny
    - Czasami domyślny stan jest ustalony przez Javę jest dla nas niewystarczający
    - Obiekt może potrzebować ustawić jakieś wartości lub wykonać jakiś kod
- Java dostarcza 3 sposoby ustalania początkowego stanu
    - Inicjalizacja pól
    - Konstruktory
    - Bloki inicjalizacyjne

Stan początkowy składowych klasy

- Inicjalny stan składowych klasy jest ustalany w trakcie tworzenia obiektu

byte short int long	float double	char	boolean	typy referencyjne
0	0.0	\u0000	false	null

Inicjalizacja pól (składowych)

- Polega na ustawieniu początkowych wartości podczas deklaracji pól poprzez:
  - Przypisanie wartości
  - Wykonanie działania
  - Użycie referencji do innego pola
  - Wywołanie metody

Konstruktor

- Konstruktor to kod wykonywany podczas tworzenia obiektu w celu ustalenia jego inicjalnego stanu
    - Nie ma zdefiniowanego żadnego typu zwracanego
    - Każda klasa ma przynajmniej jeden konstruktor
        - Jeśli nie ma go jawnie określonego to Java dostarcza jeden domyślny
- Klasy mogą mieć wiele konstruktorów
    - Każdy ma różną ilość parametrów
    - Jeden konstruktor może wywołać inny za pomocą słowa kluczowego *this*

Bloki inicjalizacyjne

- Bloki inicjalizacyjne to kod, który wykonuje się, gdy tworzona jest instancja klasy

- Zachowują się tak, jakby zostały umieszczone na początku każdego konstruktora

- Tworzy się je poprzez umieszczenie kodu wewnątrz nawiasów klamrowych { }, które umieszcza się wewnątrz klasy, ale poza jakąkolwiek metodą lub konstruktorem

Zadanie

- Zmodyfikuj klasy Trojkat, Prostokat, Kwadrat i Kolo utworzone w poprzednim zadaniu w taki sposób by pola klasy zostały zainicjowane w konstruktorach tych klas

Niezmienność parametrów

- Parametry są przekazywane poprzez utworzenie kopii ich wartości - takie podejście nazywamy przekazywaniem przez wartość

- Zmiany wykonane w przekazanej wartości nie są widoczne poza metodą

- Zmiany wykonane w polach przekazanej instancji klasy są widoczne poza metodą

Przesłanianie

- Klasa może posiadać wiele wersji swoich konstruktorów oraz metod. Ta „umiejętność” nazywana jest przesłanianiem
- Każdy konstruktor i każda metoda ma swoją unikalną sygnaturę, która składa się z 3 części:

1. Nazwy
2. Liczby parametrów
3. Typu każdego parametru

Zmienna ilość parametrów (varargs)

- Metoda może być zadeklarowana w taki sposób, by przyjąć zmienną liczbę parametrów

- Aby tego dokonać definiuje się typ parametru z trzema kropkami

- Można to zastosować tylko dla ostatniego parametru

- Metoda otrzymuje wartości w postaci tablicy

Dziedziczenie klas

- Klasy mogą być zadeklarowane tak, by dziedziczyły z innej klasy. Używa się do tego słowo kluczowe *extends*

- Klasa potomna przejmuje charakterystykę klasy bazowej

- Klasa potomna może dodać pewne specjalizacje

    - Pola o tej samej nazwie co pola z klasy bazowej są przesłaniane

    - Klasa może zostać przypisana do zmiennej bazowego typu referencyjnego

    - Klasa może przeciążyć metody z klasy bazowej o tej samej sygnaturze

Klasa Object

- Klasa Object jest bazą w hierarchii klas
    - Oznacza to, że każda klasa posiada charakterystykę klasy Object
    - Użyteczna gdy chcemy zadeklarować zmienne, pola lub parametry, które mogą być referencją do jakiejkolwiek klasy lub tablicy

Metody klasy Object

Metoda	Opis
clone	Tworzy nową instancję obiektu, która kopiuje obecny stan
hashCode	Na podstawie stanu obiektu wylicza liczbę całkowitą (hash), reprezentujący ten obiekt
getClass	Zwraca informację o typie
finalize	Obsługuje scenariusz czyszczenia danych
toString	Zwraca łańcuch znaków reprezentujący bieżącą instancję obiektu
equals	Porównuje inny obiekt z bieżącym pod kątem równości strukturalnej