Metody statyczne klasy

SPIS TREŚCI

Czym są metody i jaki jest cel ich stosowania?

Każdy program komputerowy jest zbiorem indywidualnych instrukcji które wykonywane są przez procesor sekwencyjnie, czyli po kolej, jedna po drugiej.

Ilość wykonywanych instrukcji w czasie działania nawet stosunkowo prostego programu jest zwykle ogromna i wszystkie te instrukcje - jedna po drugiej - muszą zostać przekazane do procesora do wykonania. Jednak znaczna część instrukcji wykonywanych przez komputer nie jest unikalna. Program wykonuje te same sekwencje poleceń wielokrotnie, a czasem nawet większość czasu zajmuje mu wykonywanie tych samych instrukcji.

Wielokrotne deklarowanie takich samych instrukcji byłoby czasem bardzo nieefektywne, a czasem w ogóle niemożliwe. Problem powtarzalności wykonywanych instrukcji programu rozwiązuje się na dwa sposoby.

- przez deklarowanie pętli zawierających bloki poleceń które mogą być wykonywane wielokrotnie w kółko

```
double [] temp = new double[1_000];

for (int i = 0; i < 1_000; i++)
```

```
{
   temp[i] = sc.nextDouble();
}
```

- oraz za pomocą funkcji czyli zbiorów instrukcji przechowywanych w określonych miejscach pamięci do którego w razie potrzeby przekazujemy argumenty, które po przetworzeniu przez te instrukcje zwracają jakiś wynik - wartość lub działanie. W ten sposób przez wykorzystanie funkcji deklarujemy w programie określony zbiór instrukcji tylko jeden raz i odwołujemy się do nich wielokrotnie w razie potrzeby. Funkcje w programowaniu można porównać do funkcji w matematyce które pobierają jakiś argument lub zbiór argumentów i zwracają jakiś wynik.

W językach obiektowych zbiór poleceń dostępnych pod wspólną nazwą, mogących przyjmować jakieś argumenty i mogące zwracać jakiś wynik, określamy nazwą **metody**. Wynika to z tego, że w programowaniu obiektowym indywidualne wartości przechowywane są w zmiennych będącymi polami klas i te pola modyfikowane są zasadniczo przez dedykowane do tego funkcje, które też są składowymi określonej klasy. (Tę łączność pól klasy i funkcji je modyfikujących nazywamy enkapsulacją). Te funkcje są właśnie **metodami modyfikowania tych wartości**, skąd pochodzi ta na pierwszy rzut oka dziwna nazwa.

Metody statyczne

Tematem tej instrukcji są metody statyczne klasy, czyli metody które nie są składowymi powoływanych obiektów, ale całej klasy i które są po prostu fragmentem sekwencyjnie wykonywanego kodu źródłowego programu który można wywołać w razie potrzeby.

Przykładem powszechnie wykorzystywanych metod statycznych wbudowanych w język Java jest metoda Math.sqrt(double argument) pozwalająca na obliczenie pierwiastka kwadratowego albo Math.random() zwracająca liczbę pseudolosową z przedziału od 0 do 1.

Przykłady metod statycznych

Poniższe deklaracje definiują metodę o nazwie pozdrowienia, która wyświetla pozdrowienie dla użytkownika programu.

```
public static void pozdrowienia()

{
     System.out.println("Witaj Drogi Użytkowniku!");
}
```

Z kolei poniższe deklaracje definiują metodę pobierającą imię użytkownika i pozdrawiając go imiennie.

```
public static void pozdrowienia(String imie)

{
     System.out.println("Witaj Drogi " + imie + " !");
}
```

Sygnatura metody

Pierwsza linia deklaracji metody określana jest mianem sygnatury. Zawiera ona słowa kluczowe, nazwę metody i opcjonalną listę argumentów wymienionych wewnątrz nawiasów okrągłych. Przykładowo statyczna metoda

```
public static void main(String[] args)
```

z wnętrza której uruchamiamy wszystkie pisane przez nas programy zawiera deklarację określające, że jest

- public czyli publicznie dostępna dla każdego,
- static czyli jest metodą statyczną którą można używać bez konieczności tworzenia obiektu,
- void czyli nie zwraca żadnej wartości,
- main to jest jej nazwa,
- () nawiasy oznaczają, że nie jest to zmienna tylko metoda,
- String[] args to lista i typ argumentów które metoda może przyjmować z miejsca wywołania, którym w tym przypadku jest konsola.

Natomiast w przypadku podanej wcześniej metody pozdrowienia() jej sygnatura jest następująca:

```
public static void pozdrowienia()
```

Czyli jak poprzednio metoda jest publiczna, czyli może być wywoływana przez każdego, jest statyczna czyli nie wymaga tworzenia obiektu klasy, nie zwraca żadnej wartości (ponieważ jedynie wyświetla komunikat) i nazywa się pozdrowienia. A nawiasy na końcu oznaczają że jest to metoda, a nie zmienna.

Wywołanie metody

Metodę statyczny wywołujemy wewnątrz macierzystej klasy podając jej nazwę oraz parę okrągłych nawiasów[1].

```
public static void main(String[] args)

{
     pozdrowienia();
}
```

Możemy także wywołać ją też z podaniem nazwy klasy macierzystej, co jest konieczne w przypadku wywołania z innej klasy.

```
public static void main(String[] args)

{
     NazwaKlasy.pozdrowienia();
}
```

Przesyłanie argumentów i zwracanie wyniku

Łatwo można zadeklarować metodę która zarówno pobiera argument z miejsca wywołania, jak i zwraca do niego obliczoną wartość. Poniższe deklaracje definiują metodę obliczającą i zwracającą kwadrat liczby przekazanej jako argument wywołania.

```
public static double kwadrat(double liczba)

{
     return liczba * liczba;
}
```

W tym przypadku metodę tę należy wywołać podając wartość liczby której pierwiastek należy obliczyć.

```
public static void main(String[] args)

{
     double wynik = kwadrat(2);
}
```

W tym przypadku sygnatura metody kwadrat zawiera słowo double - oznaczające, że metoda zwraca pojedynczą wartość typu double - oraz deklarację (double liczba) oznaczającą, że metoda do wykonania wymaga podania jednej wartości typu double.

[1] Natomiast w przypadku wywołania poza macierzystą klasą konieczne jest także podanie jej nazwy, podobnie jak ma to miejsce w przypadku metody Math.sqrt().

```
public static double kwadrat(double liczba)
```

Metody mogą przyjmować dowolną ilość argumentów - może to być zero argumentów, jeden, dwa i więcej. Poniższa metoda oblicza sumę dwóch liczb przekazanych za pomocą dwóch argumentów.

```
public static double suma(double liczba1, double liczba2)

{
     return liczba1 + liczba2;
}
```

Powyższa metodę wywołamy podobnie jak wcześniejszą, jednak w tym przypadku podamy dwa argumenty.

```
public static void main(String[] args)

{
     double wynik = suma(2, 5);
}
```

Zwracanie wielu wartości jest możliwe

Metody zwracają zawsze tylko jeden wynik, czyli pojedynczą wartość jednak ta wartość może być też referencją do obiektu dzięki czemu faktycznie można zwracać wiele wartości "zapakowanych" np. do tablicy, jak w poniższym przykładzie.

```
public static int[] rzut5Kostkami()

{
     Random r = new Random();

     return new int[]

     {
        r.nextInt(6) + 1,
        r.nextInt(6) + 1,
        r.nextInt(6) + 1,
        r.nextInt(6) + 1,
        r.nextInt(6) + 1,
        r.nextInt(6) + 1,
     }
}
```

Powyższa metoda zwraca pięcioelementową tablicę reprezentującą pojedynczy rzut pięcioma kościami w grze w kości.

Metodę wywołujemy tak samo jak poprzednio z wyjątkiem deklaracji zmiennej w której zapisywany jest wynik zwracany do wywołującego polecenia z głównego programu. Co więcej konieczne jest dodanie nawiasów prostokątnych do sygnatury metody wskazujących że zwraca ona tablicę liczb a nie pojedynczą wartość.

```
public static void main(String[] args)

{
     double[] wynik = rzut5Kostkami();
}
```

Przeciążanie

Ciekawą właściwością metod w jakie jest to, że mogą być przeciążane, co oznacza to, że można definiować wiele metod o tej samej nazwie, które jednak pobierają różne ilości argumentów i dla procesora faktycznie są różnymi metodami.

Przykładowo metoda iloczyn w przypadku podania jednego argumentu zwróci komunikat, że konieczne jest podanie ich większej ilości, a w przypadku podania dwóch lub trzech argumentów zwróci ich iloczyn.

```
public static void iloczyn(double l)

{
     System.out.println("Podaj co najmniej dwa argumenty");
}

public static double iloczyn(double l1, double l2)

{
     return l1 * l2;
}

public static double iloczyn(double l1, double l2, double l3)

{
     return l1 * l2 * l3;
}
```

Zagnieżdżanie

Na koniec warto zwrócić uwagę jeszcze na to, że metody można wywoływać z wnętrza innych metod. W gruncie rzeczy robiliśmy to od początku wywołując deklarowane metody z wnętrza uruchamianej automatycznie przez komputer metody main.

Podsumowanie

Metody są wygodnym sposobem wielokrotnego wykorzystania tych samych fragmentów kodu źródłowego programu.

Czytaj więcej

---

1. **Maszyna Turinga**
   https://www.calculemus.org/forum/1/hetm01.txt
   http://www.szwarc.net.pl/przedmioty/podstawy-programowania
   https://www.google.com/search?q=maszyna+turinga+symulator

2. **Static Methods**
   https://introcs.cs.princeton.edu/java/21function/