

Przewodnik 5 – implementacja polimorfizmu statycznego w języku JAVA na przykładzie hierarchii klas Person, Employee, Secretary i Programmer

dr inż. Łukasz Sosnowski
Akademia WIT
pod auspicjami Polskiej Akademii Nauk

1 Dodanie metody w klasie bazowej

W klasie bazowej przekazanej hierarchii klas, dodaj metodę publiczną:

`boolean matches(args)`

przyjmującą 4 parametry:
`String firstName,`
`String lastName,`
`Date birthFrom,`
`Date birthTo`

Wykonaj implementację metody w taki sposób aby:

1. jeśli dany parametr przekazany do metody ma wartość null, wtedy nie jest sprawdzany na nim żaden warunek,
2. jeśli parametr `firstName` nie jest null ani nie jest pustym łańcuchem znaków to należy sprawdzić czy stanowi podciąg znaków zmiennej składowej klasy o tej samej nazwie, lecz taki który zaczyna się od znaku o indeksie 0.
3. jeśli parametr `lastName` nie jest null ani nie jest pustym łańcuchem znaków to należy sprawdzić czy stanowi podciąg znaków zmiennej składowej klasy o tej samej nazwie, lecz taki który zaczyna się od znaku o indeksie 0.
4. jeśli parametr `birthFrom` nie jest null to należy sprawdzić czy zmienna składowa klasy `dateOfBirth` posiada wartość większą lub równą od tej przekazanej w parametrze
5. jeśli parametr `birthTo` nie jest null to należy sprawdzić czy zmienna składowa klasy `dateOfBirth` posiada wartość mniejszą lub równą od tej przekazanej w parametrze

Metoda ma zwracać `true` jeśli koniunkcja warunkowa wyrażeń logicznych zbudowanych dla powyższych punktów będzie prawdziwa.

UWAGA: jeśli wszystkie parametry przekazane do metody będą null to metoda ma zwrócić `true`.

Metoda ta ma za zadanie określać spełnienie warunku dopasowania względem przekazanych parametrów.

2 Przeciążenie metody w klasie Employee

Zdefiniuj metodę przeciążoną **matches** dla parametrów takich jak z sekcji 1 oraz dodatkowo:

Date employmentFrom,

Date employmentTo,

BigDecimal salaryFrom,

BigDecimal salaryTo

Wykonaj implementację z użyciem metody z klasy bazowej oraz dodatkowo obsłuż warunki dla zmiennych składowych zdefiniowanych w tej klasie (Employee) w analogiczny sposób jak było to opisane w sekcji 1 dla podobnych parametrów.

3 Przeciążenie metody w klasie Secretary

Zdefiniuj metodę przeciążoną **matches** dla parametrów takich jak z sekcji 2 oraz dodatkowo:

Set<String> languages

Wykonaj implementację z użyciem metody z klasy bazowej dla tej klasy oraz dodatkowo obsłuż warunek dla zbioru znanych języków obcych w analogiczny sposób jak było to opisane w sekcji 1. Dla zbioru języków wartość **true** ma zostać osiągnięta jeśli zbiór z parametrów jest podzbiorem zbioru języków zdefiniowanym w klasie Secretary.

4 Przeciążenie metody w klasie Programmer

Zdefiniuj metodę przeciążoną **matches** dla parametrów takich jak z sekcji 2 oraz dodatkowo:

Map<String,Short> mapProgrammingLanguages

Wykonaj implementację z użyciem metody z klasy bazowej dla tej klasy oraz dodatkowo obsłuż warunek dla znanych języków programowania wraz z oceną ich znajomości w analogiczny sposób jak było to opisane w sekcji 1. Dla zbioru języków programowania wartość **true** ma zostać osiągnięta jeśli zbiór języków z parametru jest podzbiorem zbioru języków zdefiniowanym w klasie Programmer i każdy z języków jest znany na co najmniej takim samym poziomie co przekazane w parametrze.