Programowanie obiektowe

Laboratorium 10 - klasy abstrakcyjne i interfejsy

mgr inż. Krzysztof Szwarc

krzysztof@szwarc.net.pl

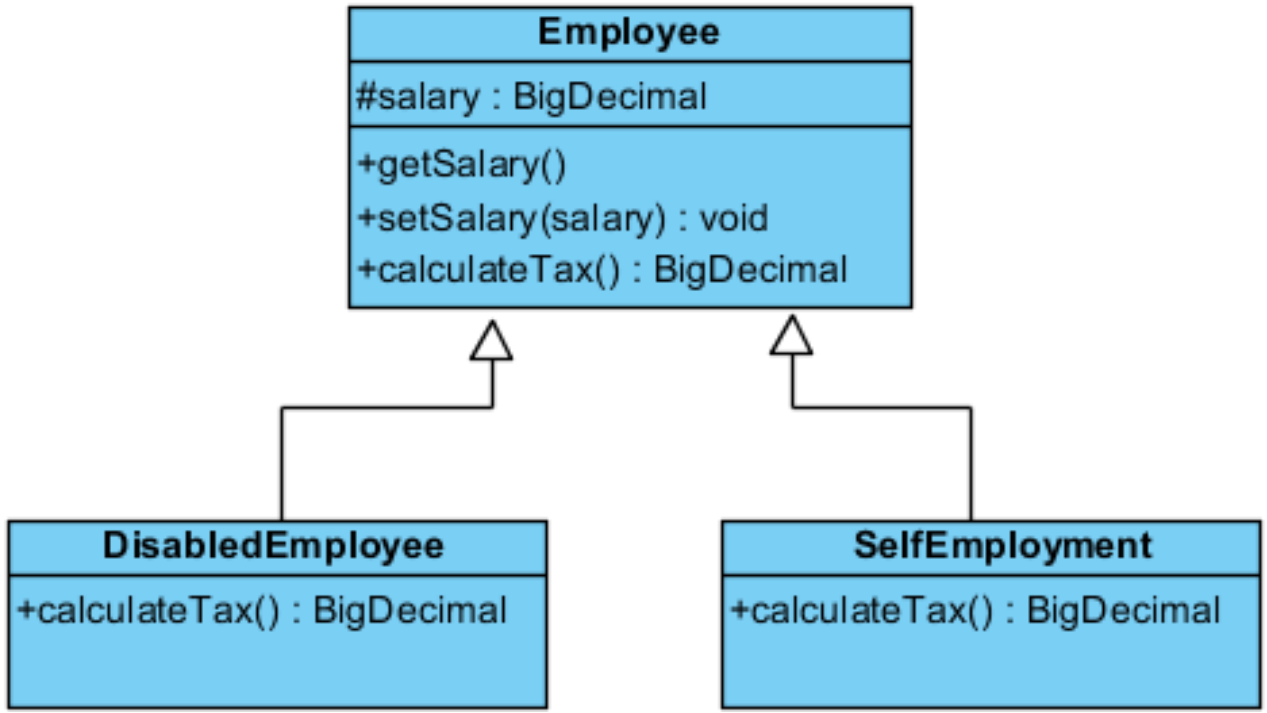
Sosnowiec, 17 maja 2017

Czym jest polimorfizm?

Definicja

Polimorfizm (z gr. wielopostaciowość) jest mechanizmem pozwalającym na wyabstrahowanie wyrażeń od konkretnych typów - obiekt może przyjąć jedną z wielu przeznaczonych dla niego postaci, cechujących się odmiennym zachowaniem. Jest on jednym z czterech, najważniejszych założeń paradygmatu obiektowego (abstrakcja, hermetyzacja, polimorfizm oraz dziedziczenie).

Praktyczne zastosowanie polimorfizmu




```
Employee[] employees = {new Employee(), new
    DisabledEmployee(), new SelfEmployment()};
for (Employee employee : employees)
    System.out.println(employee.calculateTax());
```

Czym jest klasa abstrakcyjna?

Definicja

Klasa abstrakcyjna (ang. abstract class) jest klasą, której obiektów nie da się utworzyć - jest wyłącznie uogólnieniem innych klas.

Język	Sposób utworzenia
$C++$	Ma min. 1 metodę czysto wirtualną
$C\#/Java/PHP$	Za pomocą słowa kluczowego *abstract*

Język	Sposób użycia
$C++/C\#$	klasaPochodna : klasaBazowa
$Java/PHP$	Za pomocą słowa kluczowego *extends*

Przykład klas abstrakcyjnych

C++

```
class Abstrakcyjna
{
    public:
        int pole;
        virtual void metodaAbstr() = 0;
        void metodaZImplementacja() {};
};
```

C++

```
class Potomna : public Abstrakcyjna
{
    public:
        void metodaAbstr(){
            // implementacja
        }
};
```

C#

```
public abstract class Abstrakcyjna
{
    public int pole;
    public void MetodaZImplementacja() {}
    public abstract void MetodaAbstr();
}
```

C#

```
public class Potomna : Abstrakcyjna
{
    public override void MetodaAbstr() {
        // implementacja
    }
}
```

Java

```
public abstract class Abstrakcyjna
{
    public int pole;
    public void metodaZImplementacja() {}
    public abstract void metodaAbstr();
}
```

Java

```
public class Potomna extends Abstrakcyjna
{
    @Override
    public void metodaAbstr() {
        // implementacja
    }
}
```

PHP

```
abstract class Abstrakcyjna
{
    public $pole;
    public function metodaZImplementacja() {}
    abstract public function metodaAbstr();
}
```

PHP

```
class Potomna extends Abstrakcyjna
{
    public function metodaAbstr(){
        // implementacja
    }
}
```

Czym jest interfejs?

Definicja

Interfejs (ang. interface) jest abstrakcyjnym typem niezawierającym pól, które mogą zmieniać wartość i zwykle nie posiadającym implementacji zadeklarowanych metod (np. od Javy 8 możliwe jest użycie metod domyślnych - słowo kluczowe *default*). Gdy klasa definiuje wszystkie metody interfejsu oznacza to, że go implementuje.

Język	Sposób użycia
$C++$	Ma tylko metody czysto wirtualne
$C\#/Java/PHP$	Za pomocą słowa kluczowego *interface*

Język	Sposób utworzenia
$C++/C\#$	klasaPochodna : klasaBazowa
$Java/PHP$	Za pomocą słowa kluczowego *implements*

Przykład interfejsów

C++

```
class Interfejs
{
   public: virtual void metodaAbstr() = 0;
};
```

C++

```
class Klasa : public Interfejs
{
   public:
      void metodaAbstr(){
         // implementacja
      }
};
```

C#

```
public interface IInterfejs
{
   void metodaAbstr();
}
```

C#

```
public class Klasa : IInterfejs
{
   public void metodaAbstr(){
      // implementacja
   }
}
```

Java

```
public interface Interfejs
{
   public void metodaAbstr();
}
```

Java

```
public class Klasa implements Interfejs
{
   @Override
   public void metodaAbstr(){
      // implementacja
   }
}
```

PHP

```
interface Interfejs
{
   public function metodaAbstr();
}
```

PHP

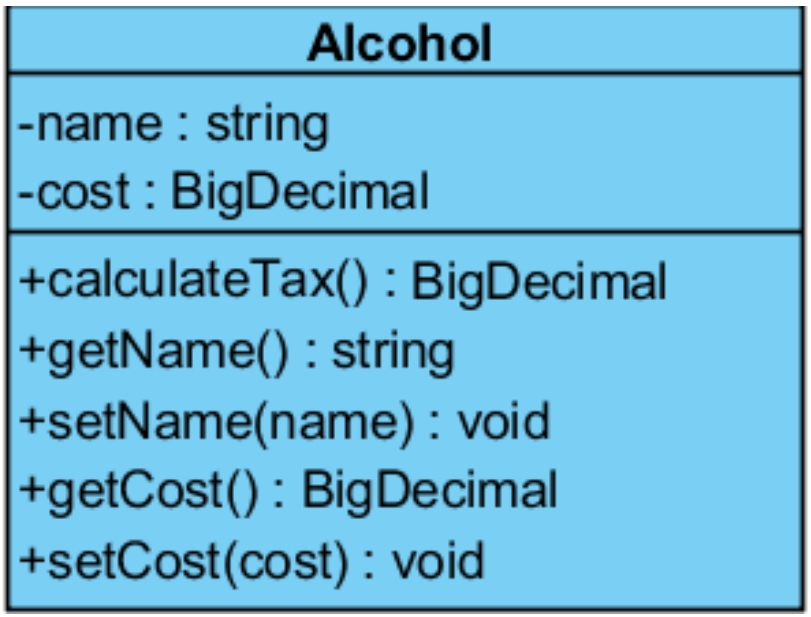
```
class Klasa implements Interfejs
{
   public function metodaAbstr(){
      // implementacja
   }
}
```

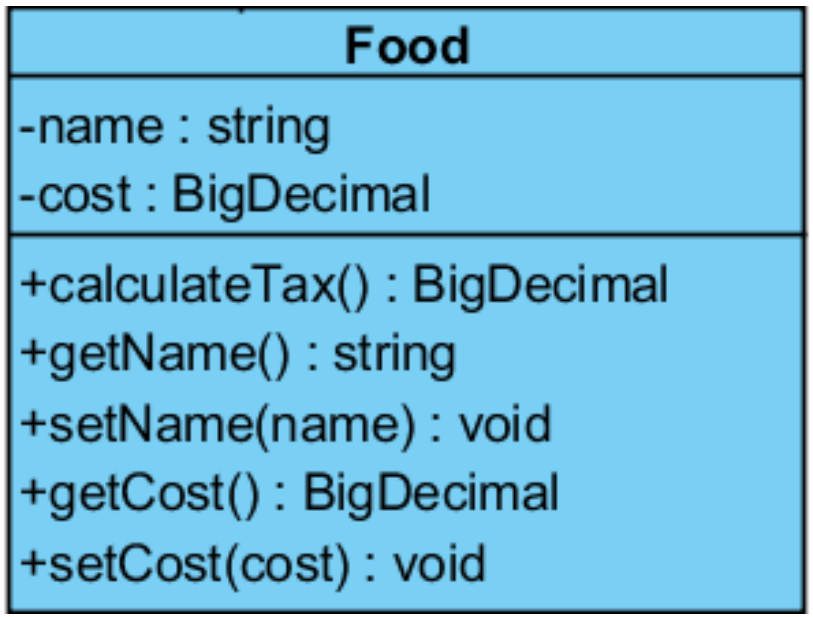
Główne różnice

1. W interfejsach wszystkie metody są abstrakcyjne (wyjątek stanowi nowy mechanizm znajdujący sie w Javie 8, jednakże nie powinien on być nadużywany), a w klasie abstrakcyjnej można stworzyć także metody posiadające ciało.
2. Klasa abstrakcyjna - w przeciwieństwie do interfejsu - powinna być związana z klasami dziedziczącymi w sensie logicznym.
3. W wielu językach wielodziedziczenie nie jest możliwe (z omawianych - $C\#$, $Java$, PHP), lecz istnieje możliwość implementowania wielu interfejsów.
4. Wszystkie zadeklarowane metody w interfejsie muszą być publiczne.
5. Interfejsy nie mogą zawierać innych atrybutów, niż stałe.

Zadanie

- Napisz dwie klasy, zgodnie z przedstawionym diagramem. Dla metody „calculateTax” skorzystaj z wysokości podatku równej 8% dla artykułów spożywczych i 23% dla alkoholu.



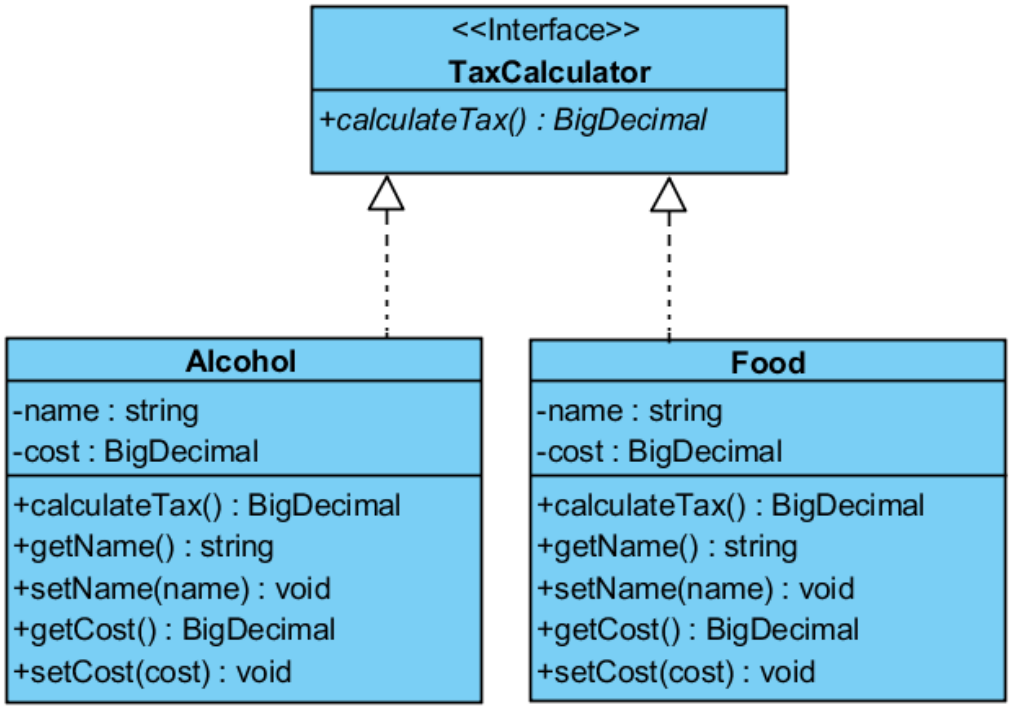


Zadanie

- Stwórz trzy produkty spożywcze i dwa alkoholowe. Oblicz sumaryczną wysokość zapłaconego podatku.

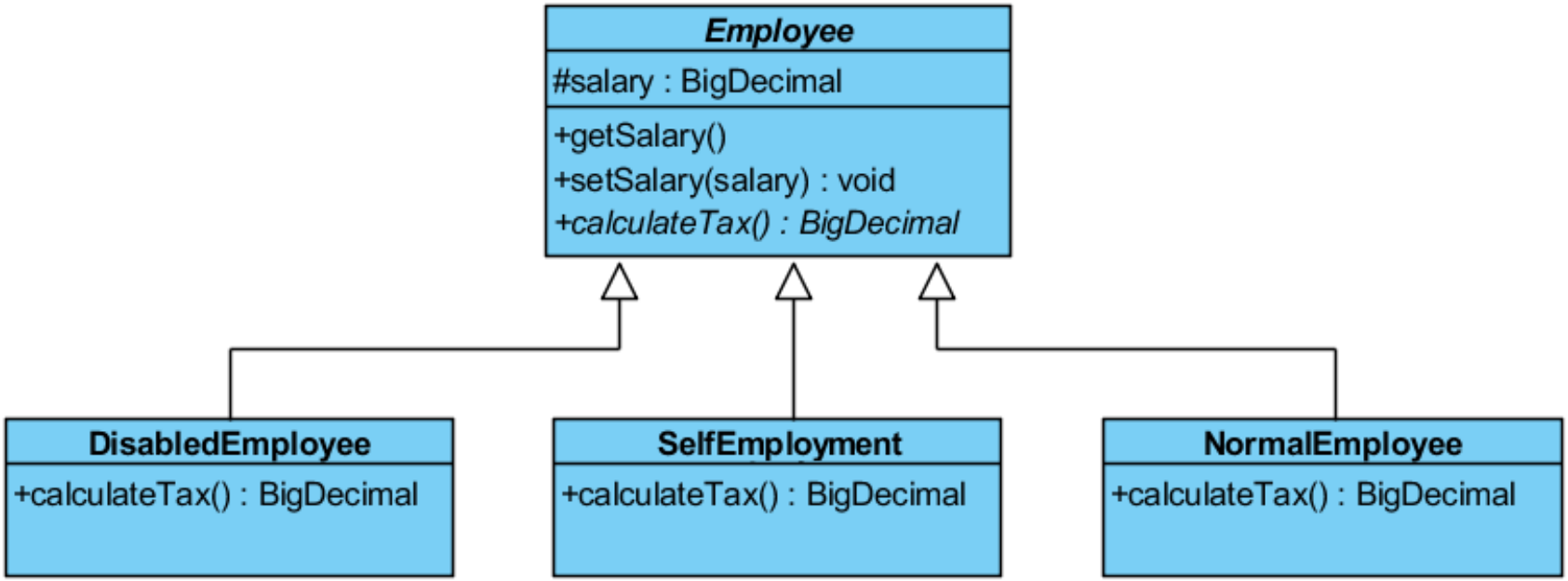
Zadanie

- Zaimplementuj interfejs zgodnie z przedstawionym diagramem i ponownie oblicz sumaryczną wysokość zapłaconego podatku. Czy moglibyśmy użyc klasy abstrakcyjnej zamiast niego?



Zadanie

- Zaimplementuj klasy zgodnie z diagramem (*Employee* jest klasą abstrakcyjną). Czy możemy zastąpić klasę *Employee* interfejsem?



Dziękuję za uwagę