Praktyczny polimorfizm

Zaawansowane metody programowania

mgr inż. Krzysztof Rewak

6 maja 2019

Wydział Nauk Technicznych i Ekonomicznych Państwowa Wyższa Szkoła Zawodowa im. Witelona w Legnicy

Plan prezentacji

- 1. Polimorfizm
- 2. Polimorfizm ad hoc
- 3. Polimorfizm uniwersalny
- 4. Podsumowanie

Polimorfizm

Czym jest polimorfizm?

Z greki *wielopostaciowość*, polimorfizm to umożliwienie jednym sposobem dostępu do wielu różnych *wartości*.

Wykorzystanie polimorfizmu może być jednym z podstawowych narzędzi refaktoryzacji kodu w celu uzyskania tzw. czystego kodu.

Wykorzystanie polimorfizmu powinno natomiast być podstawowym narzędziem przy planowaniu architektury aplikacji, zarówno na poziomie całego projektu, jak i jego części.

Polimorfizm ad hoc



Tzw. polimorfizm *ad hoc* to sposób na wywoływanie określonych funkcji na argumentach wybranego typu.

Przeciążanie operatorów

Jednym z narzędzi polimorfizmu *ad hoc* jest przeciążanie operatorów, o którym była już mowa na zajęciach z Projektowania i programowania obiektowego I.



public string hello()
 return "hello" + "world"



public Matrix recalculate(Matrix a, Matrix b, int scalar)
 Matrix t = transpose(a)
 return scalar * t * b



Area area = new Area(1000)

if(event.intercepted("increase")
 area++
 print area



bo 453 + 451

Przeciążanie operatorów

Przeciążanie operatorów jest często nazywane *składniowym lukrem*. Oznacza to, że taka konstrukcja istnieje tylko i wyłącznie dla wygody programisty.

Ale czy lukier jest zdrowy?

Przeciążanie funkcji

Niektóre języki programowania pozwalają na tworzenie wielu różnie zaimplementowanych funkcji o tej samej nazwie. Na podstawie typów przekazanych parametrów wybierana jest konkretna implementacja do wywołania.

public int area(int width)

return width * width

public double area(double width)

return width * width

public double area(double width, double heigh

public double area(double width, double height)
 return width * height

Koercja

Większość języków programowania dopuszcza tzw. rzutowanie typów. Czy wymuszoną konwersję jednego typu na drugi można nazwać implementacją polimorfizmu?

```
// opakowante i rozpakowante
Integer i = new Integer(44);
int j = i.intValue();
// koercja automatyczna
```



Czym zatem jest polimorfizm $ad\ hoc$? I dlaczego jest $ad\ hoc$?

Polimorfizm uniwersalny

Polimorfizm uniwersalny, za Wikipedią, pozwala pisać ogólne struktury danych i algorytmy, bez precyzowania na jakich dokładnie typach one operują i bez konieczności dostarczania implementacji odpowiednich dla każdego przypadku.

https://pl.wikipedia.org/wiki/Polimorfizm_(informatyka)

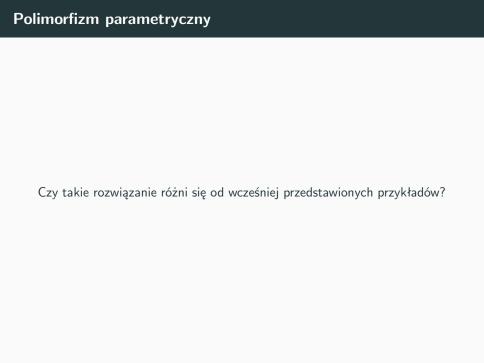
Polimorfizm parametryczny

Niektóre języki programowania pozwalają na tworzenie tzw. typów generycznych, które pozwalają na wykorzystywanie danych bez względu na ich typ, a zależnie od zaimplementowanego interfejsu.

```
public class Entry<KeyType, ValueType> {
   private final KeyType key;
   private final ValueType value;
   public Entry(KeyType key, ValueType value) {
```

thts.key = key;
thts.value = value;
}
public KeyType getKey() {
 return key;
}
public ValueType getValue() {
 return value;
}
public String toString() {

return "(" + key + ", " + value + ")";





Jak można wykorzystać polimorfizm do refaktoryzacji kodu?

```
• • •
class Bird {
  public function getSpeed() {
    switch ($this->type) {
      case EUROPEAN:
        return $this->getBaseSpeed();
      case AFRICAN:
        return $this->getBaseSpeed() - $this->getLoadFactor() * $this->numberOfCoconuts;
      case NORWEGIAN_BLUE:
        return ($this->isNailed) ? 0 : $this->getBaseSpeed($this->voltage);
    throw new Exception("Should be unreachable");
```

przykład za

https://refactoring.guru/replace-conditional-with-polymorphism

```
abstract class Bird {
  abstract function getSpeed(): float;
class European extends Bird {
 public function getSpeed(): float {
    return $this->qetBaseSpeed();
class African extends Bird {
  public function getSpeed(): float {
    return $this->qetBaseSpeed() - $this->qetLoadFactor() * $this->numberOfCoconuts;
class NorwegianBlue extends Bird {
 public function getSpeed(): float {
    return ($this->isNailed) ? 0 : $this->getBaseSpeed($this->voltage);
$speed = $bird->getSpeed();
```

Polimorfizm parametryczny

Powinniśmy stwierdzić, że drugie rozwiązanie nie tylko ładnie obrazuje metodę *Tell, Don't Ask*, ale także spełnia warunek *Open/Closed Principle*.

Podsumowanie

Bibliografia i ciekawe źródła

- www.ii.uni.wroc.pl/~zs/Dydaktyka/TPJP/JavaPolimorfizm.pdf
- https://en.wikipedia.org/wiki/Parametric_polymorphism
- https://refactoring.guru/
 replace-conditional-with-polymorphism



Kod prezentacji dostępny jest w repozytorium git pod adresem https://bitbucket.org/krewak/pwsz-zmp



Wszystkie informacje dot. kursu dostępne są pod adresem http://pwsz.rewak.pl/kursy/10

