AspectJ

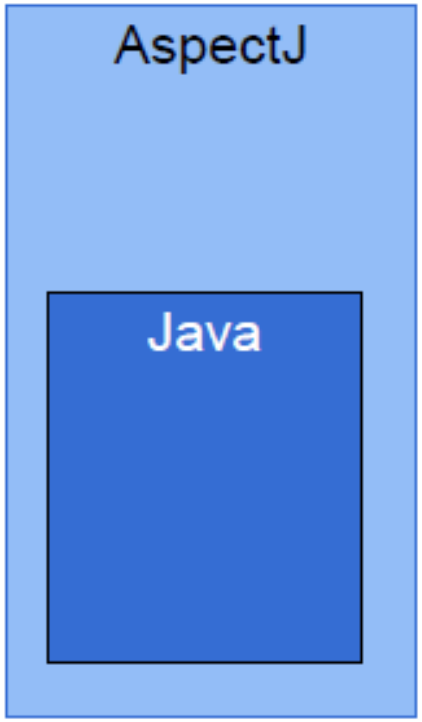
Agata Wójcik, Mateusz Winiarski

Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
AGH University of Science and Technology

31.05.2017

AspectJ

- uniwersalne aspektowe rozszerzenie Javy
- aspekt jako specyficzna klasa
- możliwość zmiany zachowania i struktury kodu
- łączenie aspektów i klas na poziomie bajtkodu
- własny kompilator *ajc, debugger ajdb*
- integracja z Eclipse IDE



Programowanie obiektowe a aspektowe

Programowanie obiektowe

- grupowanie podobnych koncepcji za pomocą hermetyzacji i dziedziczenia
- podstawowa jednostka modularyzacji: klasa

Programowanie aspektowe

- grupowanie podobnych koncepcji w niezwiązanych ze sobą klasach
- dodatkowy mechanizm modularyzacji: aspekt

Punkty złączeń

Punkty złączenia są dowolnymi, identyfikowalnymi miejscami w programie

- wywołanie metody i konstruktora
- dostęp do pola w klasie
- statyczna inicjacja
- obsługa wyjątku

Punkty przecięcia

Punkt cięcia jest zdefiniowaną kolekcją punktów złączenia.

Podział względem złożoności:

- proste punkty przecięcia
- punkty złożone

Podział względem tożsamości:

- nazwane punkty przecięcia
- anonimowe punkty przecięcia

Punkty cięcia

- call call(* Klasa.metoda*(int,..))
- execution execution(* Klasa.metoda(..))
- handler handler(RuntimeException+)
- within within(Klasa)
- withincode withincode(Klasa.m*(..))

Porady

Porada jest fragmentem kodu programu wykonywanym przed, po lub zamiast osiągnięcia przez program punktu cięcia.

Rodzaje porad:

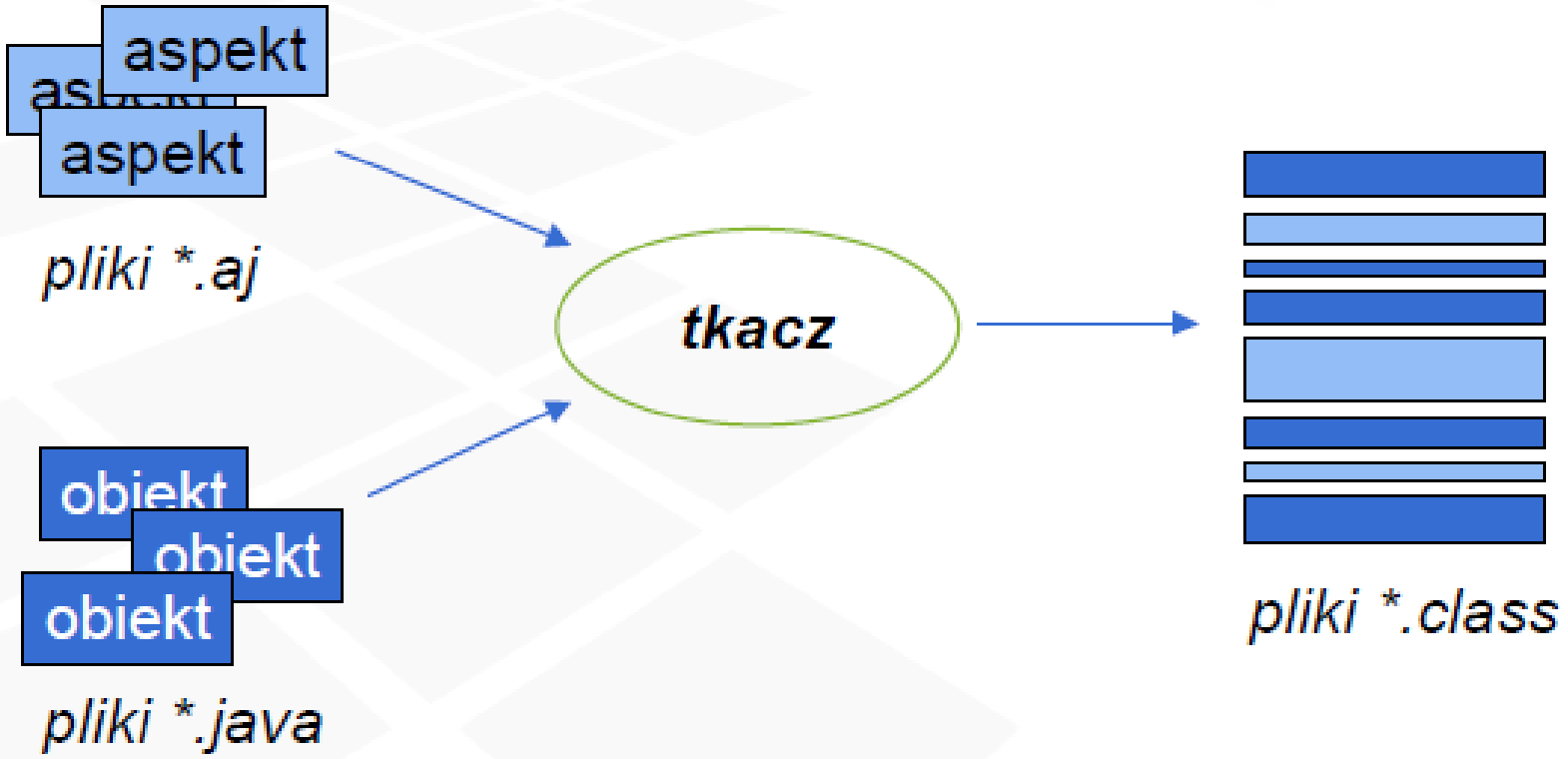
- before()
- after() {returning/throwing}
- around()

Aspekty

Aspekt (w języku AspectJ)

- specjalizowana klasa, która może przecinać inne klasy
- podlega dziedziczeniu
- jednostka modularyzacji (obok klasy)
- posiada typowe elementy klasy
- łączy punkty cięcia i porady

Tkacz

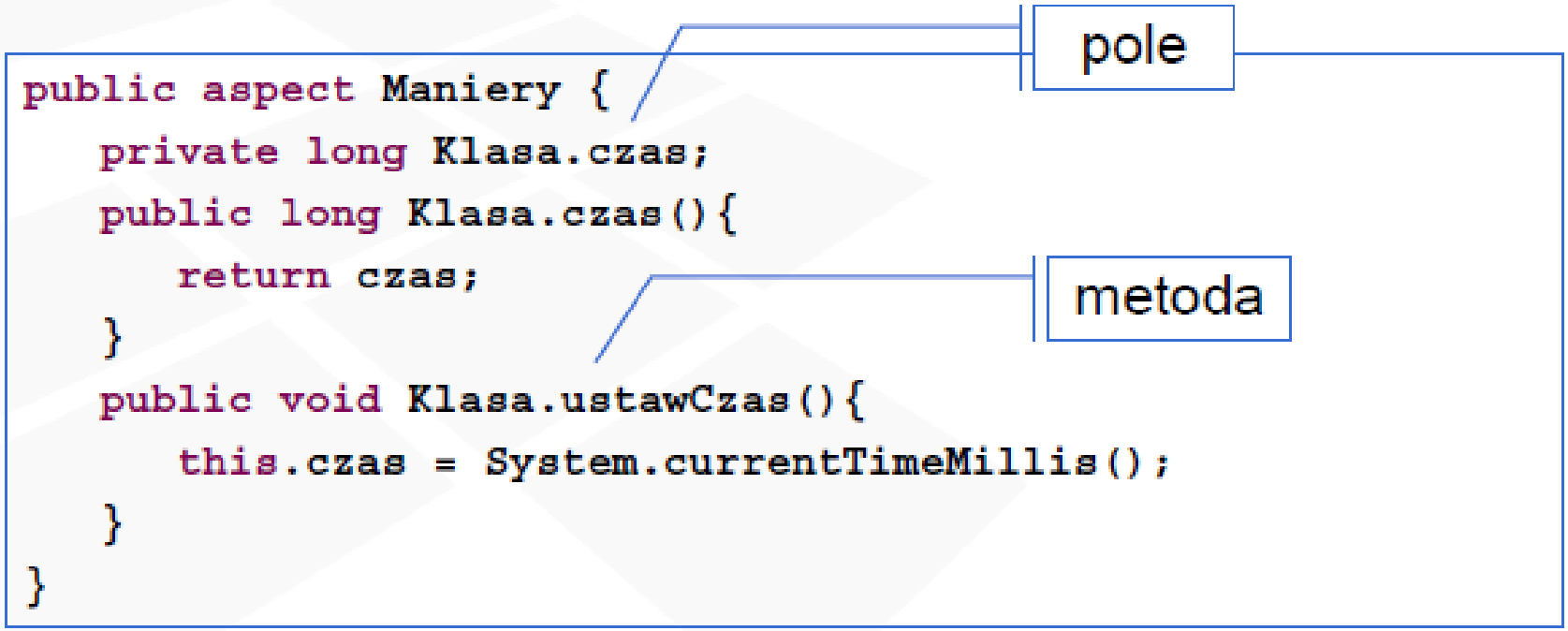

aspekt
aspekt
pliki *.aj
tkacz
obiekt
obiekt
obiekt
pliki *.java
pliki *.class

Przykład

```
public aspect Logger {
  pointcut wywolanieMetody() :
    call(* * (..)) && ! within(Logger);

  before() : wywolanieMetody() {
    System.out.println("Przed wywołaniem metody "
      + thisJoinPoint.getSignature());
  }
  after() returning : wywolanieMetody() {
    System.out.println("Po wywołaniu metody "
      + thisJoinPoint.getSignature());
  }
  after() throwing  : wywolanieMetody() {
    System.out.println("Metoda " +
      thisJoinPoint.getSignature() + " zgłosiła wyjątek");
  }
}
```

Przykład




```
public aspect Maniery {
    private long Klasa.czas;
    public long Klasa.czas(){
        return czas;
    }
    public void Klasa.ustawCzas(){
        this.czas = System.currentTimeMillis();
    }
}
```