Programowanie i projektowanie obiektowe

Klasy w Javie

Paweł Daniluk

Wydział Fizyki

Jesień 2011

Organizacja projektu w Javie

Klasy i interfejsy

Każda klasa (lub interfejs) umieszczana jest w osobnym pliku (z rozszerzeniem .java).

Pakiety

Pliki z klasami mogą być umieszczane w drzewiastej strukturze analogicznej do katalogów w systemie plików.

Modyfikatory dostępu

Dostępność elementów

Modyfikator	Wewnątrz klasy	W innej klasie w tym samym pakiecie	Podklasa w innym pakiecie	Dowolna klasa w innym pakiecie
`private`	tak	nie	nie	nie
domyślnie	tak	tak	nie	nie
`protected`	tak	tak	tak	nie
`public`	tak	tak	tak	tak

Kapsułkowanie

Często opłaca się deklarować atrybuty z modyfikatorem `private` i udostępniać metody do pobierania i zmieniania ich wartości.

Przykład

```
public class A {
    private int val;
    private boolean cond;

    int getVal() {
        return val;
    }

    void setVal(int val) {
        this.val = val;
    }

    boolean isCond() {
        return cond;
    }
}
```

Dziedziczenie

```
class Animal {
    String talk() { return "?!?!?"; }
}

class Cat extends Animal {
    String talk() { return "Meow!"; }
}

class Dog extends Animal {
    String talk() { return "Woof!"; }
}
```

Dziedziczenie c.d.

```
static void main() {
    Animal a=new Cat();
    Dog d=new Dog();

    System.out.println(a.talk());

    a=d;                              //Takie przypisanie jest ok.

    Cat c=a;                          //A takie nie.

    if(a instanceof Cat) {            //Za to wolno tak.
        Cat c = (Cat) a;
    }

}
```

Dziedziczenie c.d.

Przysłanianie metod

Jeżeli w podklasie jest zdefiniowana metoda o takiej samej nazwie jak w nadklasie, to dla każdego obiektu podklasy będzie ona wykonywana, niezależnie od typu referencji, która wskazuje na obiekt.

Anotacja oznaczająca metodę przysłaniającą:
`@Override`

Odwoływanie się do elementów nadklasy

`super` – oznacza referencję do nadklasy
`this` – oznacza referencję do samej siebie

Przykład

```
class KolorowyKlocek extends Klocek {
    String kolor;

    public String toString() {
        return super.toString()+" koloru "+kolor;
    }

    setKolor(String kolor) {
        this.kolor=kolor;
    }
}
```

Konstruktory

Konstruktor

```
class Osoba {
    String imię;
    String nazwisko;
    int wiek;

    Osoba(String imię, String nazwisko) {
        this.imię = imię;
        this.nazwisko = nazwisko;
    }

    Osoba(String imię, String nazwisko, int wiek) {
        this(imię, nazwisko);
        this.wiek = wiek;
    }
}
```

Wywołanie innego konstruktora musi być pierwszą instrukcją.

Konstruktory c.d.

Konstruktor, a dziedziczenie

Definiując konstruktor podklasy można posłużyć się konstruktorem nadklasy.

Konstruktor, a dziedziczenie

```
class Student extends Osoba {
    int nrIndeksu;

    Student(String imię, String nazwisko, int nrIndeksu) {
        super(imię, nazwisko);
        this.nrIndeksu = nrIndeksu;
    }
}
```

Przeciążanie

Przeciążanie metod

W klasie mogą być zdefiniowanych wiele metod o tej samej nazwie różniących się liczbą i typem argumentów.

```
class Kalkulator {

    int dodaj(int a, int b) { ... }

    double dodaj(double a, double b) { ... }
}
```

Modyfikatory

Modyfikatory klas

`abstract` – Klasa służy wyłącznie jako węzeł w hierarchii klas. Nie można tworzyć obiektów należących do niej.
`final` – Nie można dziedziczyć z klas oznaczonych tym atrybutem.
`static` –

Modyfikatory metod

`abstract` – Stosowany z klasach abstrakcyjnych. Oznacza, że metoda o takiej nazwie i argumentach musi być zdefiniowana we wszystkich podklasach.
`final` – Nie można przesłaniam metod oznaczonych tym atrybutem.
`static` –

Zadanie 1 – Testy

Zadanie

Tworząc odpowiednie przykłady przetestuj działanie mechanizmów przedstawionych na wykładzie.

Wskazówka

Konstrukcje niepoprawne (np. próby dostępu do atrybutów prywatnych) po przetestowaniu zasłoń komentarzem.

Zadanie 2 – Figury geometryczne

Zadanie

Utwórz hierarchię klas służącą do przechowywania informacji o figurach geometrycznych (kwadrat, prostokąt, koło, trójkąt) pozwalającą na wykonywanie następujących operacji (tam gdzie to możliwe):

- obliczanie obwodu i pola,
- obliczanie długości najdłuższego boku,
- obliczanie promienia okręgu opisanego na figurze,
- wypisywanie informacji.

Zadanie 2 – Figury geometryczne c.d.

Wskazówki

- Obliczanie promienia okręgu opisanego najłatwiej zrealizować za pomocą interfejsów, które pojawią się na następnym wykładzie.
- Każdy obiekt ma zdefiniowaną metodę `public String toString()`, która jest między innymi wykorzystywana w poleceniu `System.out.println(Object o)`.
- Funkcje matematyczne (np. pierwiastek) są metodami statycznymi w klasie Math (np. `double Math.sqrt(double d)`)