Programowanie obiektowe

Interfejsy

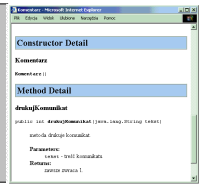
Paweł Rogaliński
Instytut Informatyki, Automatyki i Robotyki
Politechniki Wrocławskiej

pawel.rogalinski@pwr.wroc.pl

Interfejs programistyczny i jego implementacja

Interfejs programistyczny klasy to sposób komunikowania się z jej obiektami, który jest opisany przez zestaw metod (i ewentualnie atrybutów), dostępnych do użycia z poziomu innej klasy. Standardowym sposobem opisu interfejsu programistycznego jest dokumentacja generowana przez program ***javadoc***.
Implementacja klasy to konkretne definicje wszystkich metod opisanych w interfejsie programistycznym oraz wszystkich innych metod i atrybutów o dostępie prywatnym będących składowymi klasy.

```
/**
 *  To jest przykładowa klasa <code>Komunikat</code>
 *  zawierająca komentarze <i>javadoc</i>.
 *  @author Paweł Rogaliński
 *  @version v1.0 (2006r.)
 */
class Komunikat
{
    /**
     *  metoda drukuje komunikat.
     *  @param tekst treść komunikatu
     *  @return zawsze zwraca 1.
     */
    public int drukujKomunikat(String tekst)
    { System.out.println(tekst);
      return 1;
    }
}
```



Interfejsy

Interfejs w Javie to deklarowany za pomocą słowa kluczowego **interface** nazwany zbiór deklaracji zawierający:

- publiczne abstrakcyjne metody (bez implementacji),
- publiczne statyczne zmienne finalne (stałe) o ustalonych typach i wartościach.

Implementacja interfejsu w klasie polega na zdefiniowaniu w tej klasie wszystkich metod zadeklarowanych w implementowanym interfejsie.

Interfejsy cd.

Ogólna postać definicji interfejsu w języku Java:

```
public interface NazwaInterfejsu
{
   typ nazwaZmiennej = wartosc;
   ...

   typ nazwaMetody(lista_parametrów);
   ...
}
```

Uwagi:

- modyfikator dostępu **public** przed słowem **interface** może nie występować - wówczas interfejs jest dostępny tylko w bieżącym pakiecie,
- zmienne są zawsze typu **static final** i mają przypisaną wartość stałą,
- metody są zawsze abstrakcyjne (bez implementacji).

Implementacja interfejsu

Ogólna postać definicji klasy implementującej interfejs w języku Java:

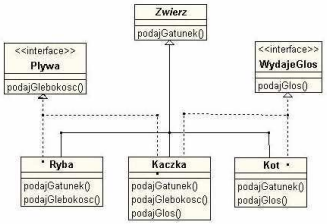
```
public class NazwaKlasy extends KlasaBazowa implements
NazwaInterfejsu_1, ..., NazwaInterjejsu_n
{
   ...
}
```

Uwagi:

- modyfikator dostępu **public** przed słowem **class** może nie występować - wówczas klasa jest dostępna tylko w bieżącym pakiecie,
- klasa może ale nie musi dziedziczyć inną klasę (słowa **extends** ***KlasaBazowa*** mogą nie występować)
- klasa może implementować wiele interfejsów,
- klasa musi definiować WSZYSTKIE metody implementowanych interfejsów albo pozostać klasą abstrakcyjną.
- klasa może zawierać własne (nie będące częścią interfejsu) atrybuty i metody.

Implementacja interfejsu – przykład

Przykład zawiera implementację klas reprezentujących różne gatunki zwierząt. Cześć zwierząt potrafi pływać, Niektóre potrafią wydawać odgłosy.



Implementacja interfejsu – przykład

```
/**
 *   Interfejs definiujący metody
 *   dla  obiektów  pływających w wodzie
 */
interface Plywa
{
    float podajGlebokosc();
}

/**
 *   Interfejs definiujący metody
 *   dla  obiektów wydających glosy
 */
interface WydajeGlos
{
    String podajGlos();
}

/**
 *   Klasa abstrakcyjna reprezentująca zwierzęta
 */
abstract class Zwierz
{   String nazwa;

    Zwierz(String nazwa){ this.nazwa = nazwa;}
    public String podajNazwe(){return nazwa; }
    public abstract String podajGatunek();
    public abstract void info();
}
```


Implementacja interfejsu – przykład cd.

Klasa **Ryba** jest klasą pochodną **Zwierz**, która implementuje interfejs **Plywa,** natomiast klasa **Kot** implementuje interfejs **WydajeGlos**

```
class Ryba extends Zwierz implements Plywa
{   float glebokosc;

    Ryba(String nazwa, float glebokosc)
       { super(nazwa);
         this.glebokosc = glebokosc;
       }
    public String podajGatunek()  {  return "Ryba";   }
    public float  podajGlebokosc(){  return glebokosc;}
    public void info()
       { System.out.println(podajNazwe() + " pływa na głębokosci " + podajGlebokosc() );
       }
}

class Kot extends Zwierz implements WydajeGlos
{
    Kot(String nazwa){ super(nazwa); }

    public String podajGatunek()  {  return "Kot";    }
    public String podajGlos()     {  return "Miauuu"; }
    public void info()
       { System.out.println(podajNazwe() + " mówi " + podajGlos());
       }
}
```


Implementacja interfejsu – przykład cd.

Klasa **Kaczka** jest klasą pochodną **Zwierz**, która implementuje zarówno interfejs **Plywa** jak i interfejs **WydajeGlos**

```
class Kaczka extends Zwierz implements Plywa, WydajeGlos
{
    Kaczka(String nazwa){ super(nazwa); }
    public String podajGatunek()  {  return "Kaczka"; }
    public float  podajGlebokosc(){  return 0.0F;     }
    public String podajGlos()     {  return "Kwa Kwa";}
    public void info()
       { System.out.println(podajNazwe() + " pływa po powierzchni i mówi " + podajGlos() );
       }
}
```


Interfejs i zmienne referencyjne

Interfejsy, podobnie jak klasy wyznaczają typy zmiennych referencyjnych.

Przykład:

Deklaracja zmiennej, której typem jest interfejs

```
WydajeGlos wg1, wg2;
```

Wartością takiej zmiennej może być odwołanie do obiektu dowolnej klasy, która implementuje ten interfejs.

```
wg1 = new Kaczka("Dziwaczka");
wg2 = new Kot("Bonifacy");
```

Przy wykonywaniu powyższych przypisań następuje referencyjna konwersja rozszerzająca referencji wskazującej na nowy obiekt do typu interfejsu (implementowany interfejs jest traktowany podobnie jak klasa bazowa).

Interfejs i zmienne referencyjne cd.

Przez zmienną której typem jest interfejs można wywołać dowolną metodę zadeklarowaną w tym interfejsie:

```
System.out.println( wg1.podajGlos() ); // wypisze "Kwa Kwa"
System.out.println( wg2.podajGlos() ); // wypisze "Miauuu"
```

Wywoła się poprawna wersja metody, odpowiednio dla faktycznego obiektu wskazywanego przez zmienną.

Wywołanie innej metody, która nie została zdefiniowana w interfejsie, jest możliwe wyłącznie po użyciu jawnej konwersji zawężającej:

```
((Zwierz)wg1).info();
      // wypisze "Dziwaczka pływa po powierzchni i mówi Kwa Kwa"

((Zwierz)wg2).info();
      // wypisze "Bonifacy mówi Miauuu"
```


Implementacja interfejsu – przykład cd.

```
class ZOO
{
    static void dialog_1(WydajeGlos z1, WydajeGlos z2)
    { System.out.print ("dialog_1: ");
      System.out.println( z1.podajGlos() + " <===> " + z2.podajGlos());
    }

    static void dialog_2(Zwierz z1, Zwierz z2)
    {
      System.out.print ("dialog_2: ");
      System.out.println( ((WydajeGlos)z1).podajGlos() + " <===> "
                        + ((WydajeGlos)z2).podajGlos() );
    }

    static void dialog_3(Zwierz z1, Zwierz z2)
    { System.out.print ("dialog_3: ");
      if (z1 instanceof WydajeGlos && z2 instanceof WydajeGlos)
          System.out.println( ((WydajeGlos)z1).podajGlos() + " <===> "
                            + ((WydajeGlos)z2).podajGlos() );
      else
          System.out.println( "  Dialog niemozliwy !" );
    }

  // ciag dalszy na nastepnej stronie
```

Zawężająca konwersja referencyjna do typu **WydajeGlos**.

Sprawdzanie czy zawężająca konwersja referencyjna do typu **WydajeGlos** jest możliwa.

Implementacja interfejsu – przykład cd.

```
// ciag dalszy klasy ZOO

    public static void main(String [] args)
    {  Kaczka kaczka = new Kaczka("Dziwaczka");
       Kaczka kaczor = new Kaczka("Donald");
       Ryba rybka = new Ryba("Zlota rybka", 0.32F);
       Kot kot = new Kot("Filemon");

       kaczka.info();
       kaczor.info();
       rybka.info();
       kot.info();

       System.out.println("\n<--- dialog_1 ---------->");
       dialog_1(kaczka, kaczor);
       dialog_1(kaczor, kot);
      //  dialog_1(kot, rybka);

       System.out.println("\n<--- dialog_2 ---------->");
       dialog_2(kaczka, kaczor);
       dialog_2(kaczor, kot);
       dialog_2(kot, rybka);

       System.out.println("\n<--- dialog_3 ---------->");
       dialog_3(kaczka, kaczor);
       dialog_3(kaczor, kot);
       dialog_3(kot, rybka);
    }
} // koniec klasy ZOO
```

Przy przekazywaniu argumentów następuje konwersja referencyjna do typu **WydajeGlos**.

Klasa **Ryba** nie implementuje interfejsu **WydajeGlos**, a więc konwersja referencyjna do typu **WydajeGlos** nie jest możliwa

Implementacja interfejsu – przykład cd.

```
// ciag dalszy klasy ZOO

    public static void main(String [] args)
    {  Kaczka kaczka = new Kaczka("Dziwaczka");
       Kaczka kaczor = new Kaczka("Donald");
       Ryba rybka = new Ryba("Zlota rybka", 0.32F);
       Kot kot = new Kot("Filemon");

       kaczka.info();
       kaczor.info();
       rybka.info();
       kot.info();

       System.out.println("\n<--- dialog_1 ---------->");
       dialog_1(kaczka, kaczor);
       dialog_1(kaczor, kot);
      //  dialog_1(kot, rybka);

       System.out.println("\n<--- dialog_2 ---------->");
       dialog_2(kaczka, kaczor);
       dialog_2(kaczor, kot);
       dialog_2(kot, rybka);

       System.out.println("\n<--- dialog_3 ---------->");
       dialog_3(kaczka, kaczor);
       dialog_3(kaczor, kot);
       dialog_3(kot, rybka);
    }
} // koniec klasy ZOO
```

```
Wiersz polecenia
C:\Test;java>java ZOO
Dziwaczka pływa po powierzchni
Donald pływa po powierzchni
Zlota rybka pływa na glebokosci 0.32
Filemon mówi Miauuu

<--- dialog_1 ---------->
dialog_1: Kwa Kwa <===> Kwa Kwa
dialog_1: Kwa Kwa <===> Miauuu

<--- dialog_2 ---------->
dialog_2: Kwa Kwa <===> Kwa Kwa
dialog_2: Kwa Kwa <===> Miauuu
dialog_2:
java.lang.ClassCastException: Ryba

<--- dialog_3 ---------->
dialog_3: Kwa Kwa <===> Kwa Kwa
dialog_3: Kwa Kwa <===> Miauuu
dialog_3:   Dialog niemozliwy !
C:\Test;java>
```