

WPROWADZENIE

Interface Definition Language (IDL)

Ważniejsze typy danych:

Typy podstawowe:

```
octet      short      long long long float
double     long double char string boolean
```

Typy ograniczone, np.:

```
string<10> imie;
```

Własne typy, np.:

```
typedef short kolor;
typedef string imie;
```

Typy strukturalne, np.:

```
struct Osoba {
    string imie;
    string nazwisko;
    long rok_urodzenia;
}
```

Tablice, np.:

```
typedef char narodowosc[2];
typedef Kolor tablica_kolorow[10][10];
```

Sekwencje:

```
typedef sequence<string> lista_nazwisk;
typedef sequence<string,6> kod_pocztowy;
typedef sequence<Osoba> lista_osob;
```

Interfejsy i metody

```
interface Osoba {
    attribute string imie;
    attribute string nazwisko;
};
interface Pracownik : Osoba {
    attribute long pensja;

    void Zwolnij();
    void ZwikszPensje(in long podstawowa, in long premia);
    long SumaChorobowego(in long rok);
};
```

Moduły

```
module mojprogram {
    ...
    interface mojinterfejs {
        ...
    };
    ...
};
module pl {
    module edu {
        module pjwstk {
            interface mojprogram {
                ...
            };
        };
    };
};
```

Wyjątki

```
interface rachmistrz {
    exception dzielenie_przez_zero { };

    double podziel(in long dzielna, in long dzielnik)
        raises(dzielenie_przez_zero);
};
```

Program idlj

```
ildj [parametry] plik.idl
```

ważniejsze parametry:

- -f{client|server|all} - generuje kod klienta, servera, obu
- -OldImplBase - generuje kod dla BOA
- -td katalog - zapisuje generowany kod w podanym katalogu

Klasy generowane przez idlj

Dla przykładowego interfejsu:

```
interface Osoba {  
    attribute string imie;  
};
```

wywołanie `idlj -fall -OldImplBase osoba.idl` wygeneruje:

- `_OsobaImplBase.java` - klasa implementująca szkielet
- `_OsobaStub.java` - klasa implementująca pinię
- `OsobaOperations.java` - interfejs zawierający deklaracje atrybutów i metod jako kod Javy
- `Osoba.java` - interfejs dziedziczący z `OsobaOperations` (tym interfejsem posługujemy się w programach Java)
- `OsobaHelper.java` - pomocnicze metod, głównie narow
- `OsobaHolder.java` - "proteza" na argumenty out i inout z IDL, które nie dają się łatwo zmapować na Jave

ĆWICZENIE

1. Utwórz plik Arytmetyka.idl I umieść w nim kod

```
interface arytmetyka {
    attribute double s1, s2, wynik;

    exception DzieleniePrzezZero { };

    void Suma();
    void Roznica();
    void Iloczyn();
    void Iloraz() raises(DzieleniePrzezZero);
};
```

2. Wywołaj polecenie:

```
idlj -fall -OldImplBase Arytmetyka.idl
```

3. Umieść wygenerowane pliki w nowym projekcie Eclipse
4. Stworzyć klasę ArytmetykaServant dziedziczącą z _arytmetykaImplBase. Zaimplementować w niej wszystkie podziedziczone metody. Zmienne z zadeklarowanych atrybutów (s1, s2, wynik) zadeklarować lokalnie.
5. Utworzyć klasę Server z metodą main. Zainicjować w niej obsługę ORB oraz utworzyć obiekt serwanta i podłączyć do ORB:

```
org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args, null);

ArytmetykaServant as = new ArytmetykaServant();
orb.connect(as);
```

6. Zapisać do pliku referencję do obiektu na ORB

```
PrintWriter out = new PrintWriter(new BufferedWriter(
    new FileWriter("ref.ior")));

out.println(
    orb.object_to_string(as) );

out.close();
```

7. Dodać kod czekający na wywołania ze strony klientów

```
java.lang.Object sync = new java.lang.Object();
synchronized (sync) {
    sync.wait();
}
```

8. Utworzyć klasę Client z metodą main. Zainicjować w niej obsługę ORB tak samo jak w serwerze

```
org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args, null);
```

9. Odczytać z pliku referencję zapisaną przez serwer

```
FileReader fr = new FileReader("ref.ior");  
BufferedReader br = new BufferedReader(fr);  
String ior = br.readLine();
```

10. Wykorzystać referencję do otrzymania zdalnego obiektu

```
org.omg.CORBA.Object obj = orb.string_to_object(ior);  
  
arytmetyka proxy = arytmetykaHelper.narrow(obj);
```

11. Od teraz można korzystać z obiektu proxy tak jakby obiekt arytmetyka był obiektem lokalnym. Możemy zawołać np.:

```
proxy.s1(1);  
proxy.s2(2);  
proxy.Sum();  
System.out.println(proxy.wynik());
```

12. Uruchomić serwer, następnie uruchomić klienta i sprawdzić czy komunikacja działa.