Programowanie i projektowanie obiektowe

Klasy i obiekty

Paweł Daniluk

Wydział Fizyki

Jesień 2011

Typy danych (w Javie)

Typy pierwotne

- typ wartości logicznych: `boolean`
- typy całkowitoliczbowe: `byte`, `short`, `int`, `long`, `char`
- typy zmiennopozycyjne: `float`, `double`

Typy danych (w Javie)

Typy pierwotne

- typ wartości logicznych: `boolean`
- typy całkowitoliczbowe: `byte`, `short`, `int`, `long`, `char`
- typy zmiennopozycyjne: `float`, `double`

Typy referencyjne

- typy klas
- typy interfejsów
- typy tablic

Typy pierwotne

Zmienna typu pierwotnego zawiera pojedynczą wartość.

```
int i;
double f=0.5;
```

typ	wartości
`boolean`	`true`, `false`
`byte`	$-128 \div 127$
`short`	$-32,768 \div 32,767$
`int`	$-2,147,483,648 \div 2,147,483,647$
`long`	$-9,223,372,036,854,775,808 \div 9,223,372,036,854,775,807$
`char`	$0 \div 255$
`float`	
`double`	

Typy referencyjne

Zmienna typu referencyjnego ma wartość `null` lub wskazuje na wartość odpowiedniego typu. Wartości typów referencyjnych mogą się zmieniać w czasie.

```
int i,j;

i = 5;
j = i;
System.out.format("i: %d j: %d\n", i, j);

j = 3;
System.out.format("i: %d j: %d\n", i, j);
```

```
i: 5 j: 5
i: 5 j: 3
```

Typy referencyjne c.d.

```
class Test {
        int val;
}

Test obI = new Test();
obI.val = 5;
Test obJ = obI;
System.out.format("obI.val: %d obJ.val: %d\n", obI.val, obJ.val

obJ.val = 3;
System.out.format("obI.val: %d obJ.val: %d\n", obI.val, obJ.val
```

```
obI.val: 5 obJ.val: 5
obI.val: 3 obJ.val: 3
```

Programowanie obiektowe

Cel

- Typy pierwotne (i tablice) są niewygodne do opisywania rzeczywistości.
- Obiekty odpowiadają rzeczywistym bytom.
- Podobne obiekty grupowane są w klasy.
- Forma i materia – klasa i obiekty.

Historia

- Simula 67 – opracowana do modelowania statków (Norsk Regnesentral)
- Smalltalk – pierwszy nowoczesny język obiektowy (dziedziczenie) (Palo Alto Research Center)
- C++, Java, Objective-C

Programowanie obiektowe c.d.

Najważniejsze cechy

- Obiekty oprócz atrybutów zawierają metody.
- Dziedziczenie
- Polimorfizm

Przykład – polimorfizm

```
abstract class Animal {
    String talk();
}

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

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

static void write(Animal a) {
    System.out.println(a.talk());
}

static void main() {
    write(new Cat());
    write(new Dog());
}
```

Klasy w Javie

```
class Pusta {
}
```

Klasy w Javie

```
class Pusta {
}
```

Atrybuty

```
class Osoba {
    String imię;
    String nazwisko;
}
```

Klasy w Javie

```
class Pusta {
}
```

Atrybuty

```
class Osoba {
    String imię;
    String nazwisko;
}
```

Metody

```
class Osoba {
    String imię;
    String nazwisko;
    String imię() { return imię; }
    String nazwisko() { return nazwisko; }
}
```

Klasy w Javie c.d.

Konstruktor

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

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

    String imię() { return imię; }
    String nazwisko() { return nazwisko; }
}
```

Zadanie 1 – Stos/kolejka

Zadanie

Zaimplementuj kolejkę/stos liczb całkowitych (typu `int`) o pojemności 100 (można użyć tablicy).

Zadanie 1 – Stos/kolejka

Zadanie

Zaimplementuj kolejkę/stos liczb całkowitych (typu `int`) o pojemności 100 (można użyć tablicy).

Wskazówki

Trzeba zaimplementować metody `void enqueue(int i)`/`int dequeue()` (dla kolejki), albo `void push(int i)`/`int pop()` dla stosu.

Zadanie 1 – Stos/kolejka

Zadanie

Zaimplementuj kolejkę/stos liczb całkowitych (typu `int`) o pojemności 100 (można użyć tablicy).

Wskazówki

Trzeba zaimplementować metody `void enqueue(int i)`/`int dequeue()` (dla kolejki), albo `void push(int i)`/`int pop()` dla stosu.

Modyfikacje

Użyj list, aby uzyskać nieograniczoną pojemność.

Zadanie 2 – Wielkie liczby

Zadanie

Zdefiniuj klasę Liczba, która przechowuje w tablicy cyfry liczby dziesiętnej. Zdefiniuj operacje wypisywania liczby, nadawania jej wartości (w postaci konstruktora przyjmującego liczbę typu `long`) oraz mnożenia przez liczbę typu `int`. W przypadku gdy w czasie mnożenia okaże się, że tablica jest za mała, procedura mnożąca powinna kopiować jej zawartość do większej. Zdefiniuj wreszcie metodę silnia, która policzy silnię zadanej jako parametr liczby typu `int`.

Wskazówki

- Długość tablicy przechowującej cyfry można podwajać, kiedy się kończy.
- Warto zrobić do tego oddzielną metodę.

Zadanie 3 – Drzewo wyszukiwania binarnego

Zadanie

Zaimplementować drzewo wyszukiwania binarnego, do którego można:

1. wkładać liczby całkowite (`void insert(int val)`),
2. sprawdzać czy i na jakiej głębokości liczba występuje w drzewie (`int search(int val)`),
3. znajdować najmniejszą i największą liczbę w drzewie (`int min()`, `int max()`),
4. znajdować największą liczbę należącą do drzewa, która jest mniejsza od zadanej (`int upper(int val)`),
5. znajdować najmniejszą liczbę należącą do drzewa, która jest większa od zadanej (`int lower(int val)`).

Wskazówki

Drzewo może być puste.