Pemrograman Berorientasi Obyek

Pertemuan 6: Overloading Operator dan Method

Overloading Method

- Meng-overload sebuah method berarti membuat lebih dari satu versi method dengan menggunakan nama yang sama, namun dengan parameter yang berbeda, baik dari segi jumlah maupun tipe

Java

```java
public class Operator {
    public int operate(int a, int b){
        return (a*b);
    }
    public double operate(double a, double b){
        return (a/b);
    }
    public static void main(String[] args){
        Operator o=new Operator();
        int x=5, y=2;
        double n=5.0,m=2.0;
        System.out.println("Hasil operate int:"+o.operate(x,y));
        System.out.println("Hasil operate double:"+o.operate(n,m));
    }
}
```

C++

```cpp
// overloading functions
#include <iostream>
using namespace std;

int operate (int a, int b)
{
  return (a*b);
}

double operate (double a, double b)
{
  return (a/b);
}

int main ()
{
  int x=5,y=2;
  double n=5.0,m=2.0;
  cout << operate (x,y) << '\n';
  cout << operate (n,m) << '\n';
  return 0;
}
```

Overloading Method pada Ruby

- Ruby tidak memiliki fasilitas untuk overloading method
- Sama halnya seperti pada multiple constructor, trik untuk melakukan overloading method pada Ruby adalah dengan menggunakan variasi jumlah parameter method

```
1  class OverloadMethod
2      def operate(*args)
3          if (args.size==0)
4              puts "Method Operate tanpa parameter"
5          elsif (args.size==1)
6              puts args[0]
7          elsif (args.size==2)
8              puts args[0]+args[1]
9          end
10     end
11 end
12
13 om=OverloadMethod.new
14 om.operate()
15 om.operate(2)
16 om.operate(3,6)
```

Method `operate` dapat menerima 0, 1, atau 2 parameter

Overloading Operator

- Meng-overload operator berarti mengubah operasi dari operator yang dimiliki oleh bahasa pemrograman
- Overloading operator merupakan salah satu bentuk *polymorphism* dimana sebuah operator dapat melakukan operasi yang berbeda, tergantung dari parameter yang diberikan
- Overloading operator bermanfaat untuk mengerjakan operasi khusus yang berlaku pada sebuah objek. Contoh:
  - Menjumlahkan/mengurangkan 2 bilangan kompleks
  - Menjumlahkan/mengurangkan jam dalam format HH:MM:SS

Overloading Operator: C++

- C++ menggunakan kata kunci `operator` lalu diikuti dengan operator yang akan di-overload
- Operator yang dapat di-overload:
  - **Operator aritmetika**: +, -, *, /, %
  - **Operator bitwise**: ^ (XOR), | (OR), & (AND), ~ (complement), << (shift left), >> (shift right)
  - **Operator logika**: ! (NOT), && (AND), || (OR)
  - **Operator increment dan decrement**: ++, --
  - **Operator compound assignment**: +=, -=, *=, /=, dsb

Contoh pada C++

Mengubah operasi penjumlahan untuk menjumlahkan tipe data bentukan user, yakni Time yang memiliki atribut `hours`, `minutes`, dan `seconds`

```
Time operator+(const Time& lhs, const Time& rhs)
{
    Time temp = lhs;
    temp.seconds += rhs.seconds;
    temp.minutes += temp.seconds / 60;
    temp.seconds %= 60;
    temp.minutes += rhs.minutes;
    temp.hours += temp.minutes / 60;
    temp.minutes %= 60;
    temp.hours += rhs.hours;
    return temp;
}
```

Overloading Operator: Ruby

- Ruby memiliki implementasi untuk overloading operator

```ruby
class OverloadOperator
    puts "========pada Fixnum========="
    a=16
    a=a>>2
    puts a
    puts a.class
    a=2
    a=a<<2
    puts a
    puts "========pada String========="
    str="Hello"
    str=str<<"World"
    puts str
    str=str>>"World"
    puts str
end
```

Output:

```
========pada Fixnum=========
4
Fixnum
8
========pada String=========
HelloWorld
C:/Users/Intan/workspace/PBO/OverloadOperator.rb:14:in `<class:OverloadOperator>': undefined method `>>' for "HelloWorld":String (NoMethodError
        from C:/Users/Intan/workspace/PBO/OverloadOperator.rb:1:in `<main>'
```

Note: Operasi untuk operator “>>” belum ada untuk tipe data String

Overloading Operator: Ruby

- Kita coba untuk meng-overload operator “>>“ pada class String

```
class String
  def >>(value)
    self.replace(value+self)
  end
end

str="Hello"
puts str>>"World."
```

Output: World.Hello

- Operator “>>“ telah di-overload untuk dapat menampilkan proses sisip di depan untuk tipe data String

Overloading Operator: Ruby

- Perhatikan class Time berikut (modifikasi dari contoh C++ sebelumnya):

```ruby
class Time
    attr_accessor :hours, :minutes, :seconds
    def initialize(hh,mm,ss)
        @hours=hh
        @minutes=mm
        @seconds=ss
    end
    def to_s
        str=@hours.to_s+":"+@minutes.to_s+":"+@seconds.to_s
        puts str
    end
    def +(t)
        output=Time.new(0,0,0)
        output.seconds=@seconds+t.seconds
        output.minutes=@minutes+output.seconds/60
        output.seconds=output.seconds%60
        output.minutes=output.minutes+t.minutes
        output.hours=@hours+output.minutes/60
        output.minutes=output.minutes%60
        output.hours=output.hours+t.hours
        return output
    end
end
```

Tester:

```ruby
t1=Time.new(8,10,37)
t2=Time.new(2,15,30)
t3=Time.new(0,0,0)
t3=t1+t2
t3.to_s
```

Overloading Operator: Java

- Java tidak memiliki implementasi untuk overloading operator
- Satu-satunya implementasi yang mirip dengan overloading operator adalah operasi “+” pada tipe int dan String
- Contoh:

```
1  package samples;
2
3  public class OperatorOverloading {
4      public static void main(String[] args){
5          int a=5;
6          int b=6;
7          int c=a+b;
8          String s1="Hello";
9          String s2="World";
10         String s3=s1+s2;
11         System.out.println("Hasil penjumlahan integer:"+c);
12         System.out.println("Hasil penjumlahan String:"+s3);
13
14     }
15 }
```

Output:

```
Hasil operasi penjumlahan a+b=11
Hasil operasi penjumlah s1+s2=HelloWorld
```

Latihan Overloading Fungsi

- Buat class untuk sebuah vektor 3 dimensi, beri nama `Vector3`, yang memiliki atribut bertipe **array of integer** yang terdiri atas 3 elemen yang mewakili dimensi x, y, dan z.
- Dalam class tersebut, buat dua method untuk melakukan operasi perkalian (*). Ada 2 macam perkalian pada vektor:
  - Perkalian vektor dengan skalar, menghasilkan sebuah vektor. Contoh: [1 2 3] * 5 = [5 10 15]
  - Perkalian vektor dengan vektor menghasilkan sebuah bilangan skalar. Contoh: [1 2 3] * [4 5 6] = 1*4+2*5+3*6 = 32
- Method yang perlu Anda buat:
  - Konstruktor untuk menginisialisasi atributnya
  - Method perkalian vektor dengan skalar, buat dengan *return value*
  - Method perkalian vektor dengan vektor, buat dengan *return value*