Pemrograman Berbasis Obyek

Pembuatan Kelas

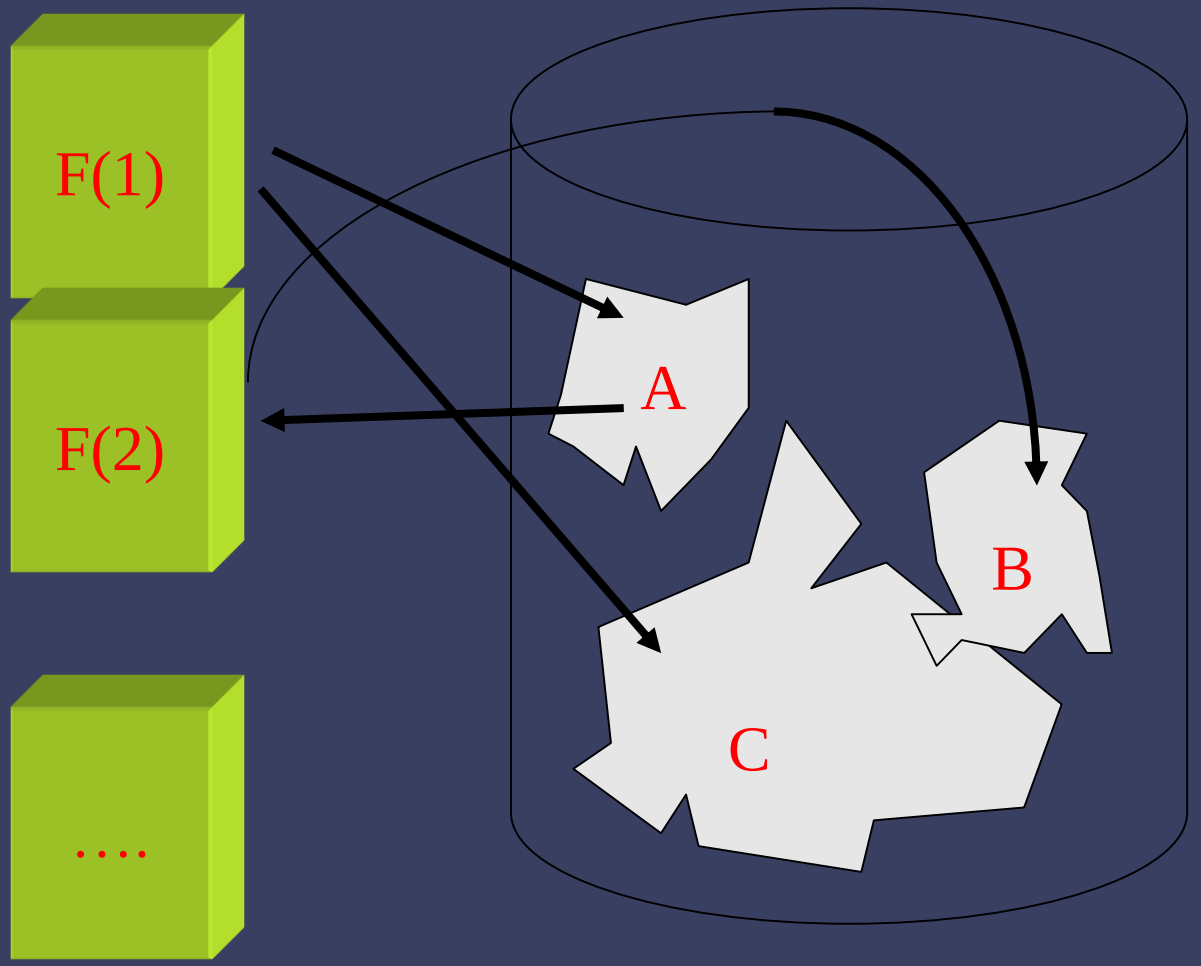
Oleh Politeknik Elektronika Negeri Surabaya
2017

Politeknik Elektronika Negeri Surabaya
Departemen Teknik Informatika dan Komputer

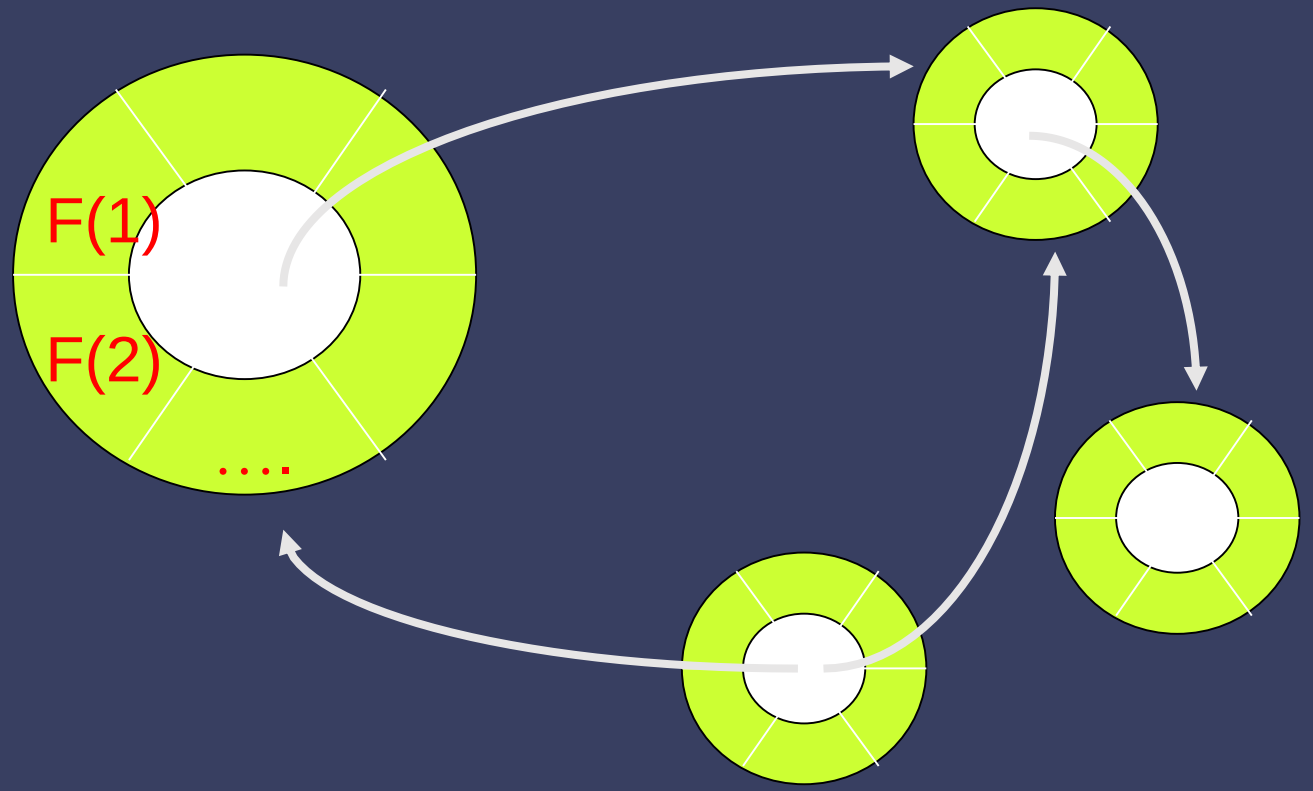
Konten

- Review konsep OOP Vs Terstruktur
- Pembuatan class
  - Deklarasi class, atribut, method dan konstruktor.
  - Membuat obyek dengan menggunakan operator new.
  - Mengakses member class.
  - State-state pada saat assign nilai pada variabel bertipe class
  - Pass by value
- Lain-lain
  - Menggunakan komentar di file sumber.
  - Variable scope.
  - Main method.
  - Java Programming Language Coding Conventions.
  - Garbage collection.

Pemrograman terstruktur



Pemrograman berorientasi obyek



Keuntungan OOP

- Reusabilitas
- Pembangunan program lebih cepat
- Fleksibilitas lebih tinggi
- Ekstensibilitas
- Less maintenance

Kata kunci OOP

- **Objek**
  Dapat berupa Class atau Instances.
  Harus berasal dari entitas atau konsep dunia nyata.
- **Atribut**
  Identitas unik dari obyek.
- **Method**
  Fungsi untuk pengaksesan atribut atau tugas tertentu.
- **Enkapsulasi**
  Menyembunyikan struktur data dan implementasi dari obyek lain.
- **Inheritansi**
  Merepresentasikan keterhubungan struktural antar obyek.
- **Polymorphism**
  Kemampuan untuk merepresentasikan 2 bentuk yang berbeda

Fitur OOP

- Encapsulation
- Inheritance
- Polymorphism

File Sumber

- File sumber Java harus diakhiri dengan ekstensi `.java`.
- Tiga top-level elemen dalam file java:
  - Package Declaration
  - Import Statements
  - Class Definitions

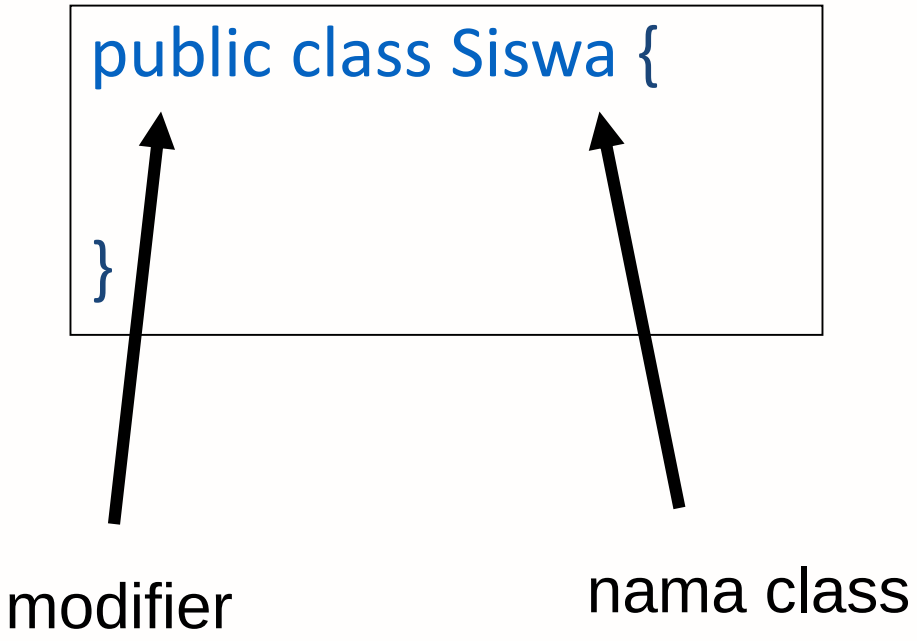
Hal yang secara implisit ada setelah kelas dikompile

- package
- import java.lang.*
- extends Object
- default constructor

Deklarasi class

```
<modifier> class <classname> {
        [deklarasi_atribut]
        [deklarasi_konstruktor]
        [deklarasi_method]
}
```

Contoh



Deklarasi Atribut

```
<modifier> <tipe> <nama_atribut>;
```

Contoh

```
public class Siswa {
    public int nrp;
    public String nama;
}
```

Deklarasi method

```
<modifier> <return_type> <nama_method> ([daftar_argumen]){
        [<statement>]
}
```

Contoh

```
public class Siswa {
    public int nrp;
    public String nama;
    public void Info() {
        System.out.println("Saya siswa PENS");
    }
}
```

Pengaksesan anggota obyek

- Gunakan notasi titik (.) sebagai berikut:

  <object>.<member>

- Member bisa berupa:
  - atribut
  - method

Contoh

```
public class IsiData {
        public static void main(String args[ ]) {
                Siswa IT2=new Siswa();
                IT2.nrp=5;
                IT2.nama=”Andi”;
                IT2.Info();
        }
}
```

Constructor (Konstruktor)

- Constructor (Konstruktor) adalah kode yang pertama kali dijalankan pada saat pembuatan suatu obyek.
- Ciri-ciri konstruktor:
  - Mempunyai nama yang sama dengan nama kelas
  - Tidak mempunyai return type
  - Memiliki argumen sebanyak 0..n

Contoh Konstruktor

```
public class Siswa{
   private int nrp;
   private String nama;

   public Siswa(int n, String m){
         nrp = n;
         nama = m;
   }
}
```

Default Constructor

- Jika tidak menuliskan kode konstruktor, maka secara otomatis kompiler akan menambahkan default constructor
- Default constructor → no argument and no body.
- The default constructor has the same access modifier as the class itself, either: public, protected, private or package (no modifier)

Contoh Default Constructor

```
modifiers ClassName() {
   super();
}
```

```
public class Siswa{
   public Siswa(){
         super(); // menjalankan konstruktor parent
   }
}
```

Ingat!!!

- Sekali saja konstruktor dibuat / ditulis secara eksplisit, maka default konstruktor akan hilang
- Contoh:

```
public class Siswa{
   String nama;
   public Siswa(int n){
         this.mana = n;
   }
}
```

Constructor (Konstruktor)

- Dalam satu kelas diijinkan memiliki beberapa konstruktor (overloading) dengan mode akses yang berbeda.
- A constructor **can use** the access modifiers **public, protected** or **private** or have no access modifier (package access)
- A constructor **can not** use the modifiers **abstract, static, final, native, synchronized** or **strictfp.**
- **T**o prevent a class from being instantiated outside the class declaration you can create a **private** constructor.

```
public class Hope{
    protected Hope(){
            for(int i =0; i <10; i ++){
                    System.out.println(i);
            }
    }

    private Hope(int number){
            for(int i =0; i <5; i ++){
                    System.out.println(i);
            }
    }
}
```

Suatu class dapat mempunyai lebih dari 1 konstruktor dengan syarat daftar parameternya tidak boleh ada yang sama

```
public class Siswa{
    private int nrp;
    private String nama;

    public Siswa(int n){
            nrp=n;
            nama="";
    }

    public Siswa(String m){
            nrp=0;
            nama=m;
    }

    public Siswa(int n, String m){
            nrp = n;
            nama = m;
    }
}
```

Suatu class dapat mempunyai lebih dari 1 konstruktor dengan syarat daftar parameternya tidak boleh ada yang sama

```
public class Siswa{
    private int nrp;
    private String nama;

    public Siswa(int n){
            this(n,"-");
    }

    public Siswa(String m){
            this(0,m);
    }

    public Siswa(int n, String m){
            nrp = n;
            nama = m;
    }
}
```

Constructor (Konstruktor)

- A method having the same name as the class is not treated as a constructor
- Contoh:

```
public void MyClassName() {} // not a constructor
public MyClassName() {} // constructor
```

- A constructor cannot have a return type.

Reference type

- Tipe selain tipe primitif dinamakan reference type
- Reference type adalah tipe berbentuk suatu class.
- Pembuatan suatu reference type untuk mengalokasikan memori dilakukan dengan menggunakan kata kunci new XXX(). Dimana XXX adalah konstruktor dari reference type

Kejadian bila new xxx() dipanggil

- Alokasi memori: ruang untuk obyek baru dibuat di memori dan variabel-variabel diset ke masing-masing nilai default-nya (false, 0, null, dll)
- Inisialisasi nilai atribut yang diberikan secara eksplisit
- Menjalankan konstruktor
- Assignment antara atribut-atribut dengan obyek

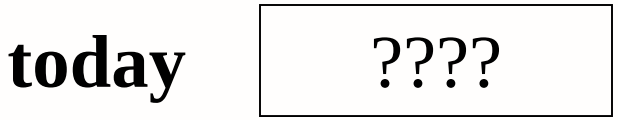
Contoh

```
public class MyDate {
        private int day=1;
        private int month=1;
        private int year=2000;
        public MyDate(int day, int month, int year) {…}
}
```

```
public class TestMyDate {
        public static void main(String args[]) {
                MyDate today=new MyDate(10,10,2005);
        }
}
```

Alokasi memori

```
MyDate today = new MyDate(10, 10, 2005);
```



Inisialisasi default value

```
MyDate today = new MyDate(10, 10, 2005);
```

today	????

day	0
month	0
year	0

Assignment nilai eksplisit

```
MyDate today = new MyDate(10, 10, 2005);
```

today	????

day	1
month	1
year	2000

Menjalankan konstruktor

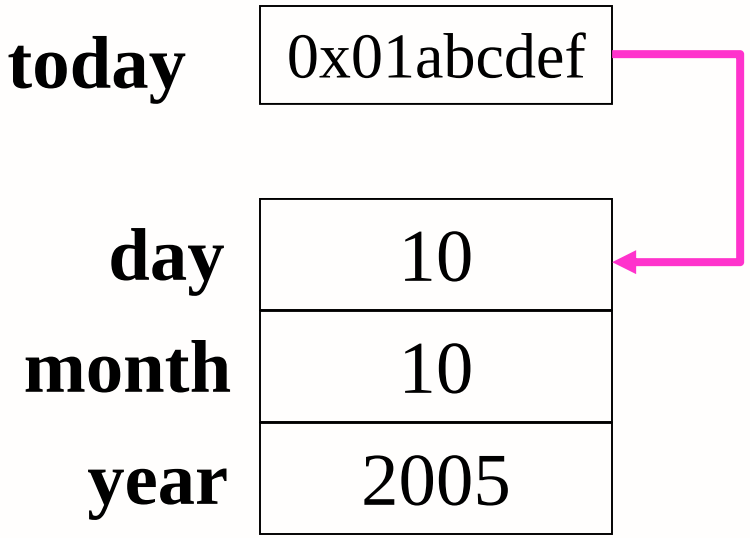
```
MyDate today = new MyDate(10, 10, 2005);
```

today	????

day	10
month	10
year	2005

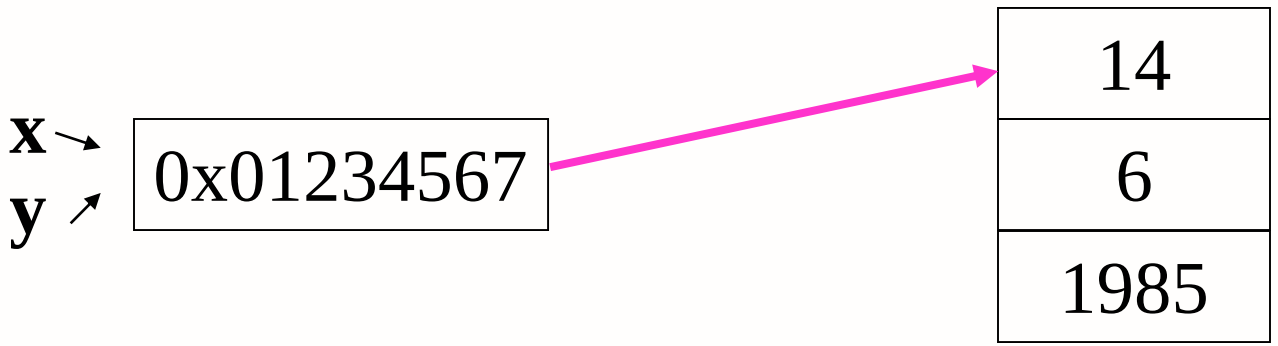
Membuat referensi

```
MyDate today = new MyDate(10, 10, 2005);
```



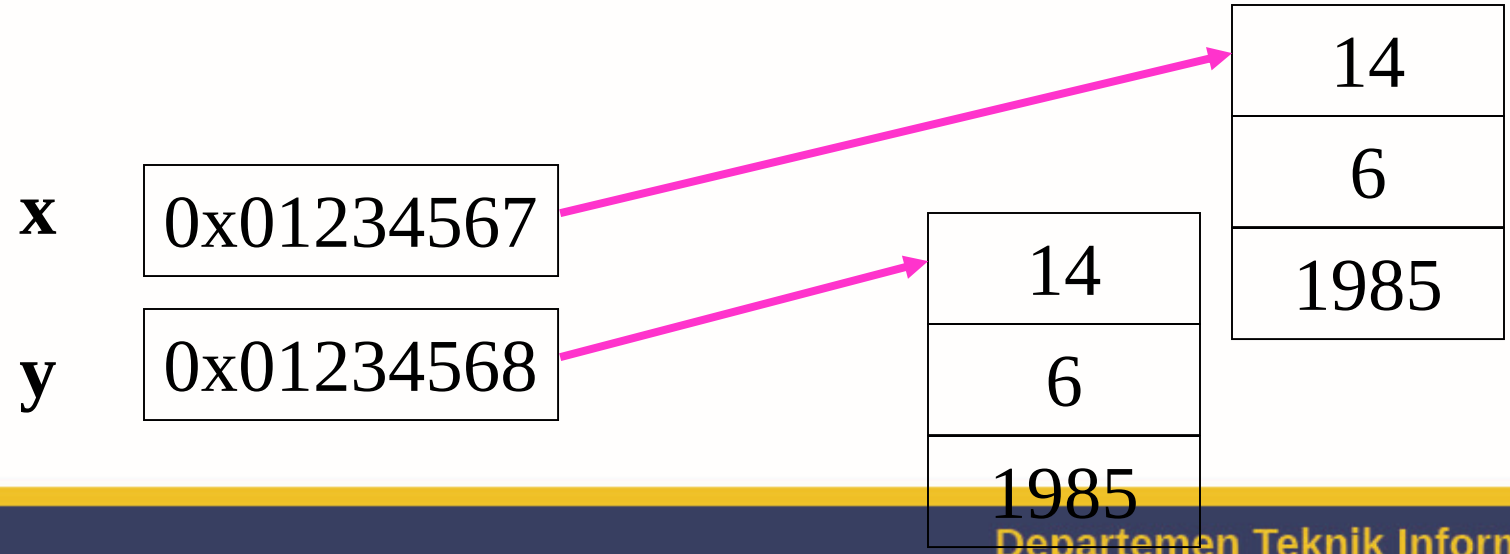
Men-assign reference variable

```
MyDate x = new MyDate(14, 6, 1985);
MyDate y = x;
```



Men-assign reference variable

```
MyDate x = new MyDate(14, 6, 1985);
MyDate y = x;
y = new MyDate(14, 6, 1985);
```



Pass by value

- Java tidak membolehkan adanya pass by reference, jadi hanya mengijinkan pass by value.
- Ketika argumen yang di-passing adalah bertipe reference type, maka anggota-anggota dari argumen tersebut diperlakukan sebagai pass by reference, sedangkan argumennya tetap sebagai pass by value

Contoh

```
public class MyDate {
        private int day=1;
        private int month=1;
        private int year=2000;
        public MyDate(int d, int m, int y) {
                day = d;
                month = m;
                year = y;
        }
        public void setDay(int d) {
                // change the day
                day = d;
        }
        public void print() {
                // print the day, month and year
        }
}
```

```
public class TestMyDate {
    public static void changeInt(int value) {
            value = 10;
    }
    public static void changeObjectRef(MyDate ref) {
            ref = new MyDate(3, 5, 2003);
    }
    public static void changeObjectAttr(MyDate ref) {
            ref.setDay(5);
    }
    public static void main(String args[]) {
            int x=5;
            changeInt(x);
            System.out.println(x);
            MyDate today=new MyDate(10,10,2005);
            changeObjectRef(today);
            today.print();
            changeObjectAttr(today);
            today.print();
    }
}
```

> java PassTest

Int value is: 5

MyDate: 10-10-2005

MyDate: 5-10-2005

Comments

- The three permissible styles of comment in a Java technology program are:

```
// comment on one line
```

```
/* comment on one
or more lines */
```

```
/** documentation comment */
```

Semicolons, Blocks, and White Space

- A *statement* is one or more lines of code terminated by a semicolon (;):

```
totals = a + b + c
         + d + e + f;
```

- A *block* is a collection of statements bound by opening and closing braces:

```
{
   x = y + 1;
   y = x + 1;
}
```

Semicolons, Blocks, and White Space

- You must use a *block* in a *class* definition:

```
public class MyDate {
   private int day;
   private int month;
   private int year;
}
```

- You can nest block statements.
- Any amount of *white space* is allowed in a Java program.

Local Variabel

- Variabel yang dideklarasikan dalam method disebut dengan variabel local, automatic, temporary, atau stack.
- Variabel yang dibuat ketika method dieksekusi dan akan dihancurkan jika keluar dari method
- Variabel yang harus diinisialisasi sebelum digunakan supaya tidak terjadi error ketika dikompile.

Variable Scope Example

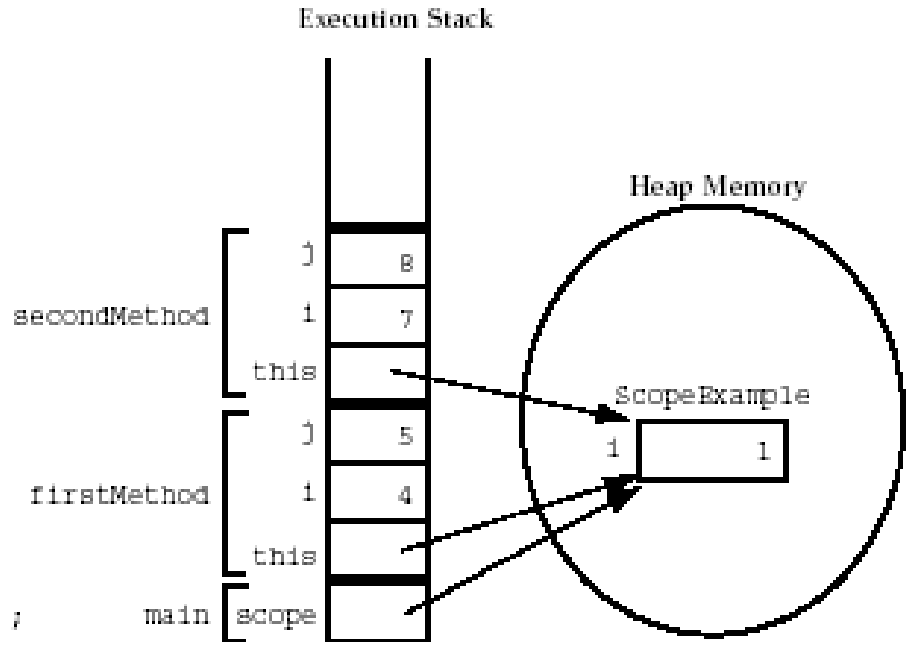
```
public class ScopeExample {
  private int i=1;

  public void firstMethod() {
    int i=4, j=5;

    this.i = i + j;
    secondMethod(7);
  }
  public void secondMethod(int i) {
    int j=8;
    this.i = i + j;
  }
}

public class TestScoping {
  public static void main(String[] args) {
    ScopeExample scope = new ScopeExample();

    scope.firstMethod();
  }
}
```



Class Fundamentals: main method

- The *main()* Method

  `public static void main(String[] args)`

  - Public : method main() dapat diakses oleh apa saja, termasuk java technology interpreter.
  - Static : keyword ini berfungsi untuk memberi tahu kompiler bahwa method main bisa langsung digunakan dalam contex class yang bersangkutan. Untuk mengeksekusi/menjalankan method yang bertipe static, tidak diperlukan instance nya.
  - Void : menunjukkan bahwa method main() tidak mengembalikan nilai
  - Main : merupakan nama method utama dari program java
  - String [] args : Menyatakan bahwa method main() menerima single parameter yaitu args yang bertipe array. Digunakan pada saat memasukkan parameter pada saat menjalankan program.

  Contoh: `java TestGreeting args[0] args[1] …`

Java Programming Language Coding Conventions

- Packages:

```
package banking.domain;
```

- Classes:

```
class SavingsAccount
```

- Interfaces:

```
interface Account
```

- Methods:

```
balanceAccount()
```

Java Programming Language Coding Conventions

- Variables:

```
currentCustomer
```

- Constants:

```
HEAD_COUNT
MAXIMUM_SIZE
```

Garbage Collection

- Allocated memory that is no longer needed should be deallocated
- In other languages, deallocation is the programmer's responsibility
- The Java programming language provides a system-level thread to track memory allocation
- Garbage collection:
  - Checks for and frees memory no longer needed
  - Is done automatically
  - Can vary dramatically across JVM implementations
- “run the garbage collector.”
  - System.gc() and Runtime.gc()

Tugas

1. Apakah yang dimaksud dengan kelas, method, atribut dan obyek?
2. Buatlah contoh suatu kelas dan definisikan atribut dan methodnya!
3. Buatlah kode program soal no. 2 diatas!
4. Buatlah kelas yang berisi main method yang membuat obyek dari kelas yang telah dibuat di soal no. 3. Selanjutnya obyek tersebut mengakses atribut dan methodnya.

1. Oracle Java Documentation, The Java™ Tutorials, https://docs.oracle.com/javase/tutorial/, Copyright © 1995, Oracle 2015.
2. Tita Karlita, Yuliana Setrowati, Rizky Yuniar Hakkun, Pemrograman Berorientasi Obyek, PENS-2012
3. Sun Java Programming, Sun Educational Services, Student Guide, Sun Microsystems, 2001.
4. John R. Hubbard, Programming With Java, McGraw-Hill, ISBN: 0-07-142040-1, 2004.
5. Patrick Niemeyer, Jonathan Knudsen, Learning Java, O'reilly, CA, ISBN: 1565927184, 2000.
6. Philip Heller, Simon Roberts, Complete Java 2 Certification Study Guide, Third Edition, Sybex, San Francisco, London, ISBN: 0-7821-4419-5, 2002.
7. Herbert Schildt, The Complete Reference, Java™ Seventh Edition, Mc Graw Hill, Osborne, ISBN: 978-0-07-163177-8, 2007