Pemrograman Berbasis Obyek

Pengantar Class Diagram

Oleh Politeknik Elektronika Negeri Surabaya

2021

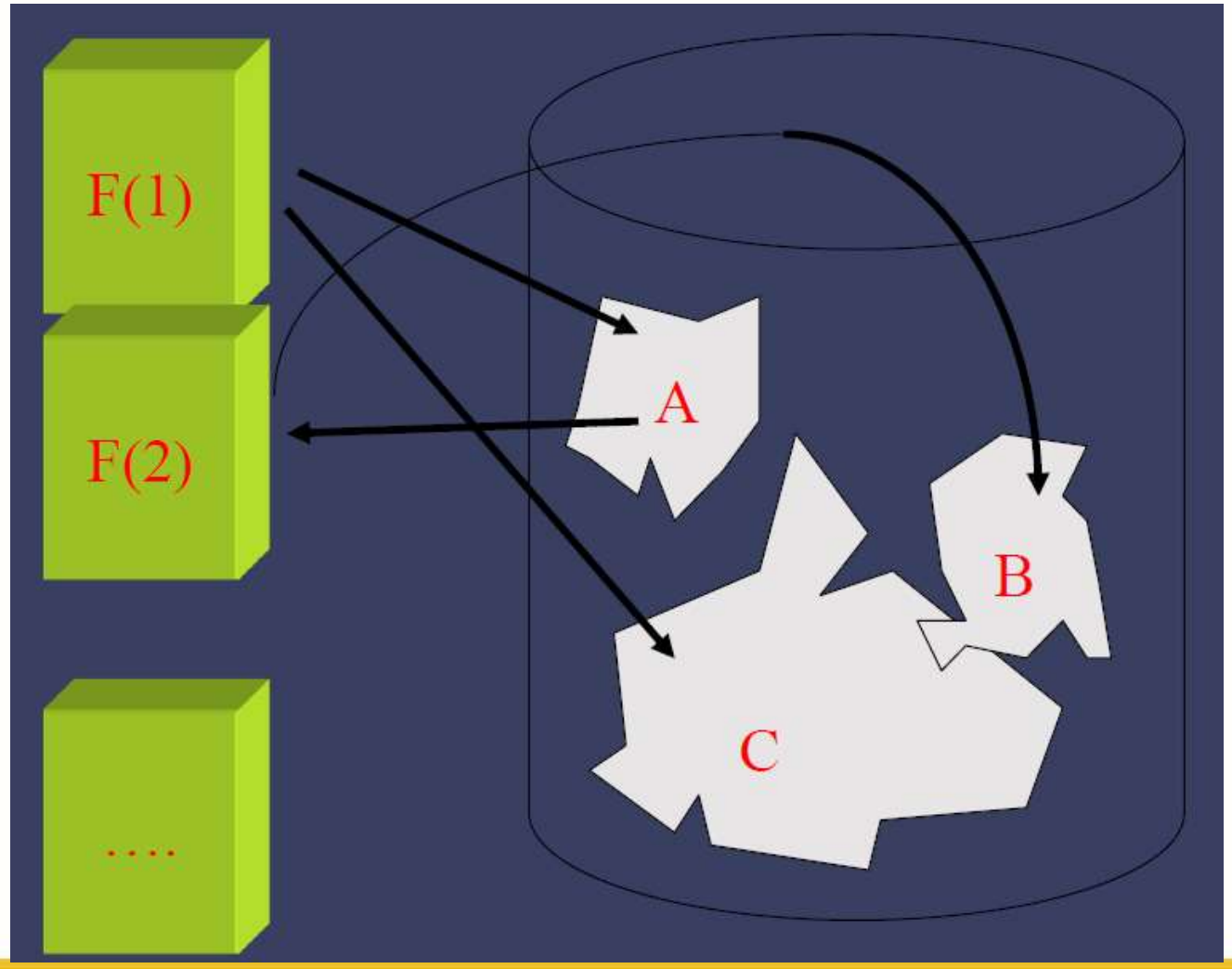
Politeknik Elektronika Negeri Surabaya
Departemen Teknik Informatika dan Komputer

Konten

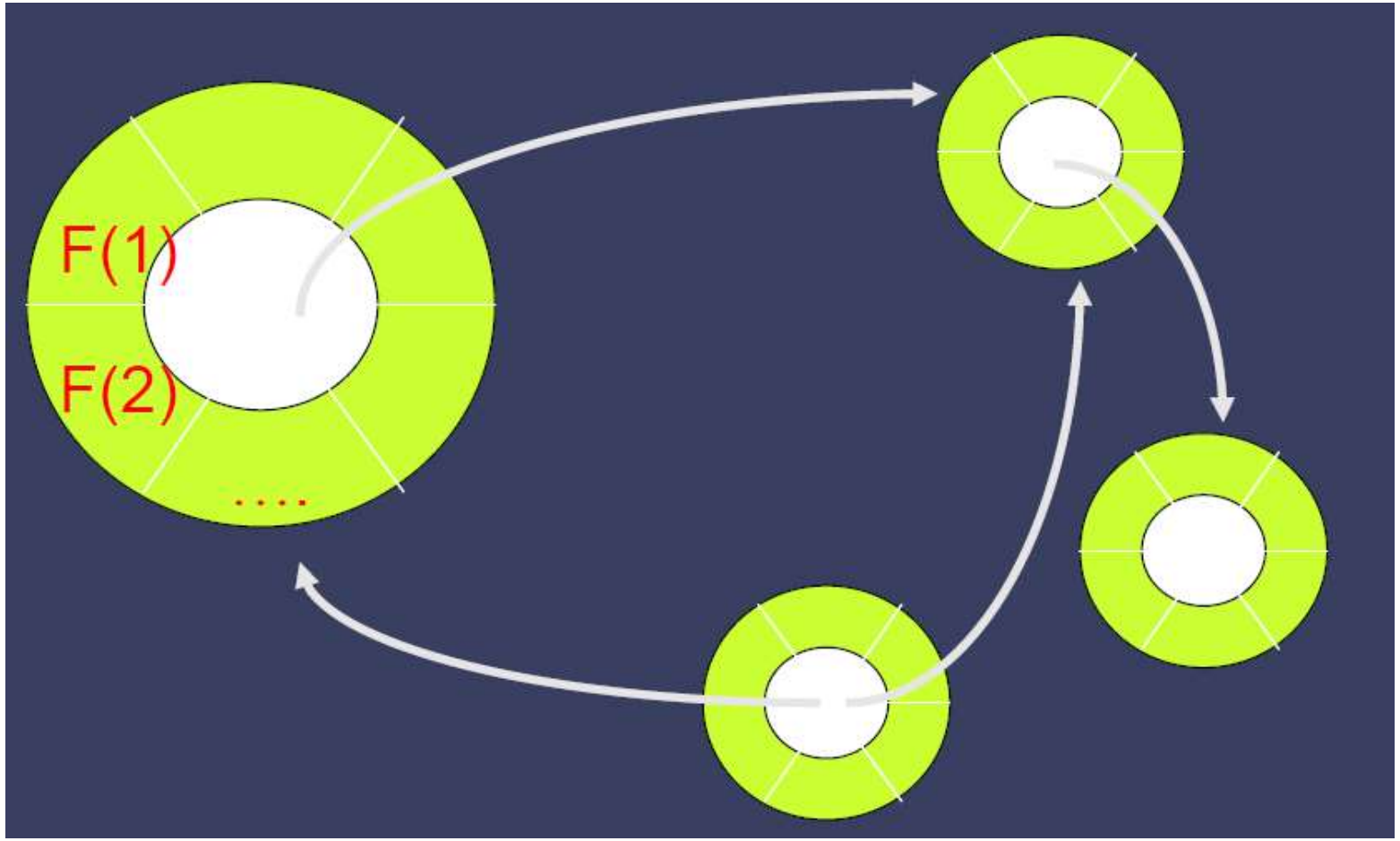
- Review Konsep OOP dan Prosedural
- Pembuatan Class
  - Deklarasi class, attribute, method
  - Pembuatan Object
  - Pengaksesan Anggota Object
  - Constructor
  - Reference type
  - Menggunakan Komentar
  - Variable scope
  - Main method
- Class Diagram

Review Konsep OOP dan Prosedural

Pemrograman Prosedural



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

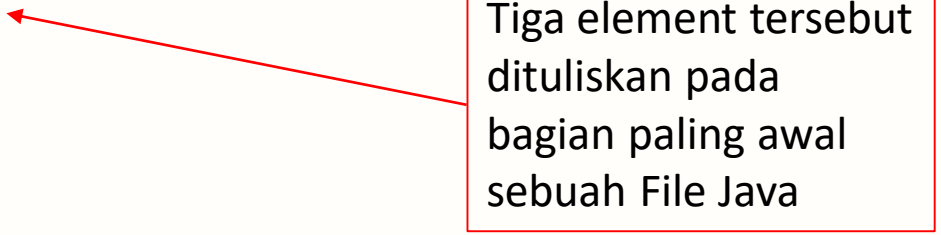
Fitur OOP

- Encapsulation
- Inheritance
- Polymorphism

Pembuatan Class

Deklarasi Class

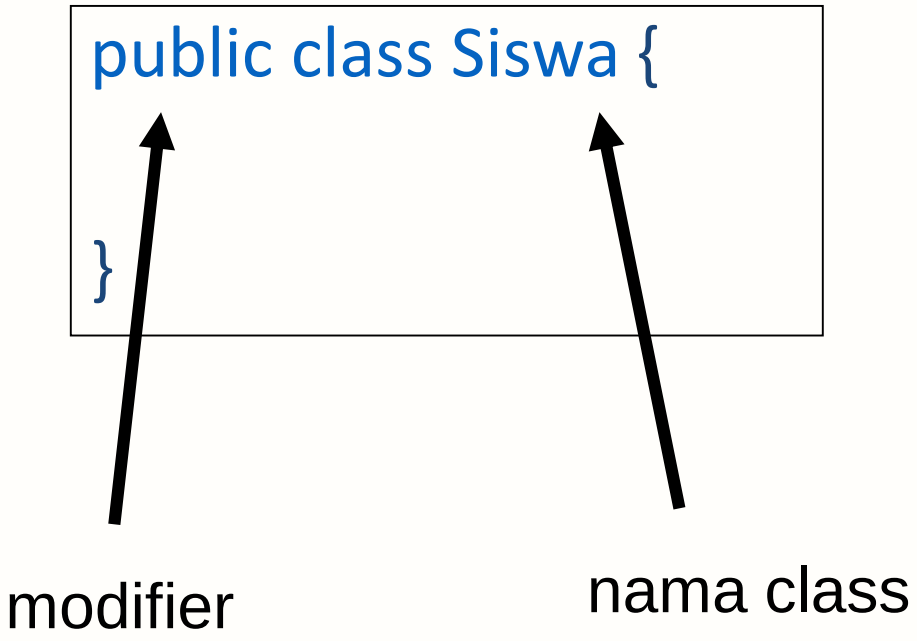
- Class merupakan salah satu Source File dalam Java
- Source File Java harus diakhiri dengan ekstensi `.java`.
- Tiga top-level elemen dalam file java:
  - Package Declaration
  - Import Statements
  - Class Definitions



Deklarasi Class

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

Contoh Deklarasi Class



Deklarasi Atribut

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

Contoh Deklarasi Atribut

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

Deklarasi method

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

Contoh Deklarasi Method

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

Pembuatan Object

- Object adalah instance dari sebuah Class
- Untuk membuat object kita harus menginstansiasi sebuah class

```
public class Siswa{
    public int nrp;
    public String nama;
    public void info(){}
}
```

```
public class IsiData{
  Public static void main(String args[]){
    Siswa budi = New Siswa();
  }
}
```

Pengaksesan Anggota Obyek

- Gunakan notasi titik (.) sebagai berikut:

  <object>.<member>

- Member bisa berupa:
  - atribut
  - method

Pengaksesan Anggota Object

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

budi adalah sebuah Object
budi adalah instance dari Class Siswa

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;
   }
}
```

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;
  }
}
```

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

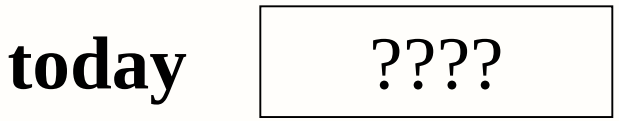
```
public class MyDate {
        private int day=1;
        private int month=1;
        private int year=2000;
        public MyDate(int day, int month, int year) {…}
}
```

```
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);
```

Sudah dibuat object today, namun belum mereference alamat memorio manapun



Inisialisasi default value

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

today	????

day	0
month	0
year	0

Menjalankan konstruktor

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

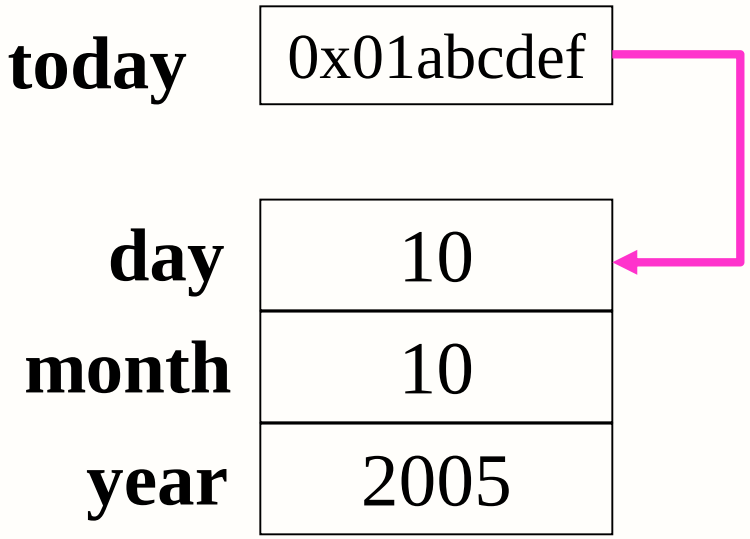
today	????

day	10
month	10
year	2005

Membuat referensi

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

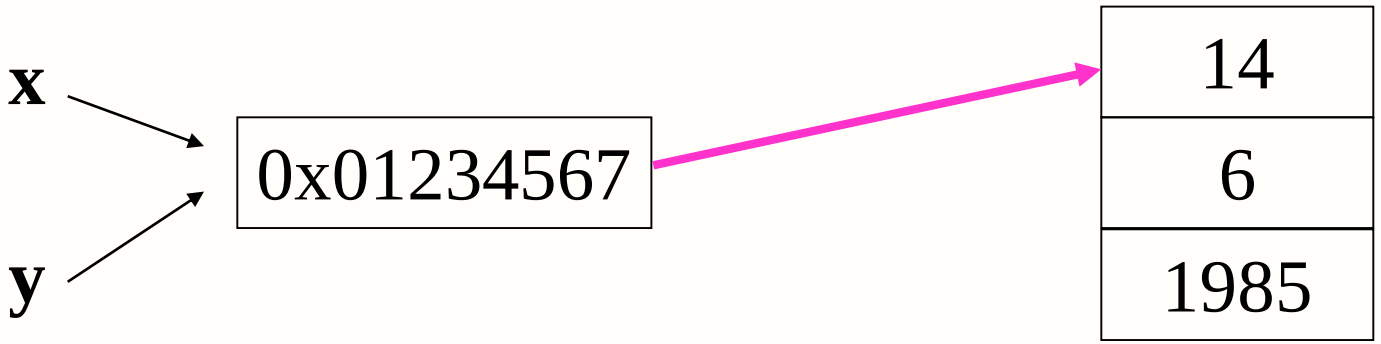
Today mereference object yang baru dibuat dalam memori



Men-assign reference variable

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

Alamat yang direferesi oleh x di assign pada y. Sehingga y dan x adalah dua object yang mereference alamat memori yang sama



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

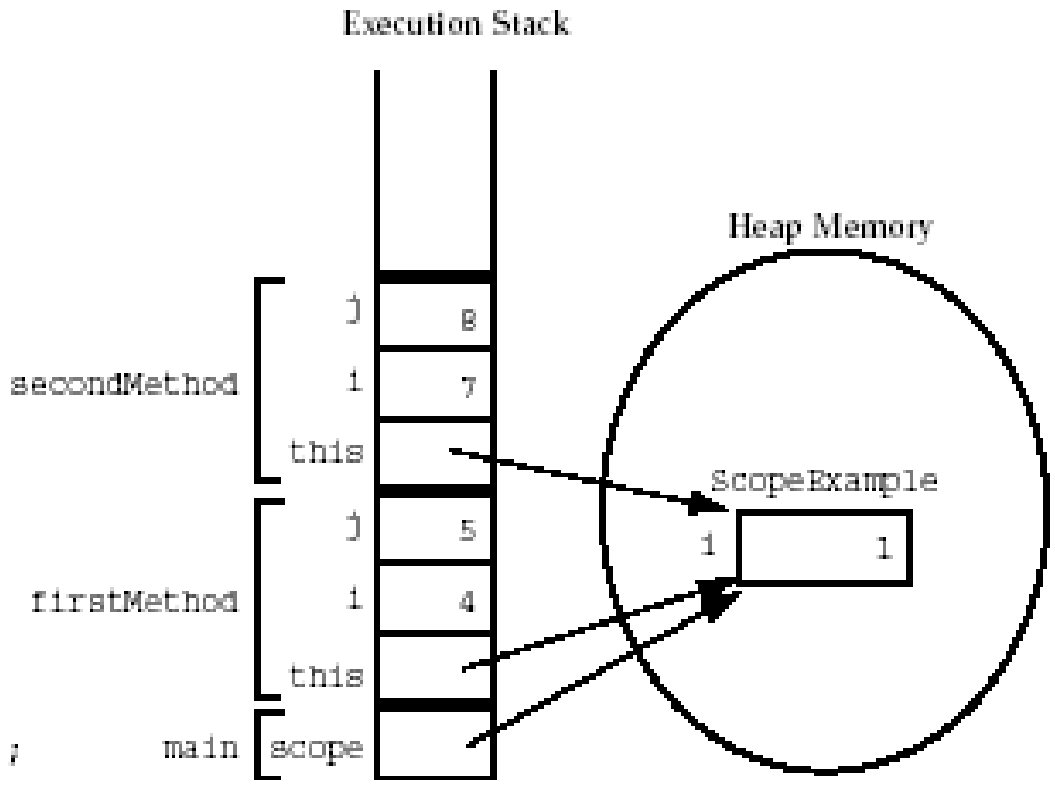
```
public class ScopeExample {
  private int i=1;

  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 Fundamental : main method

- The *main()* Method

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

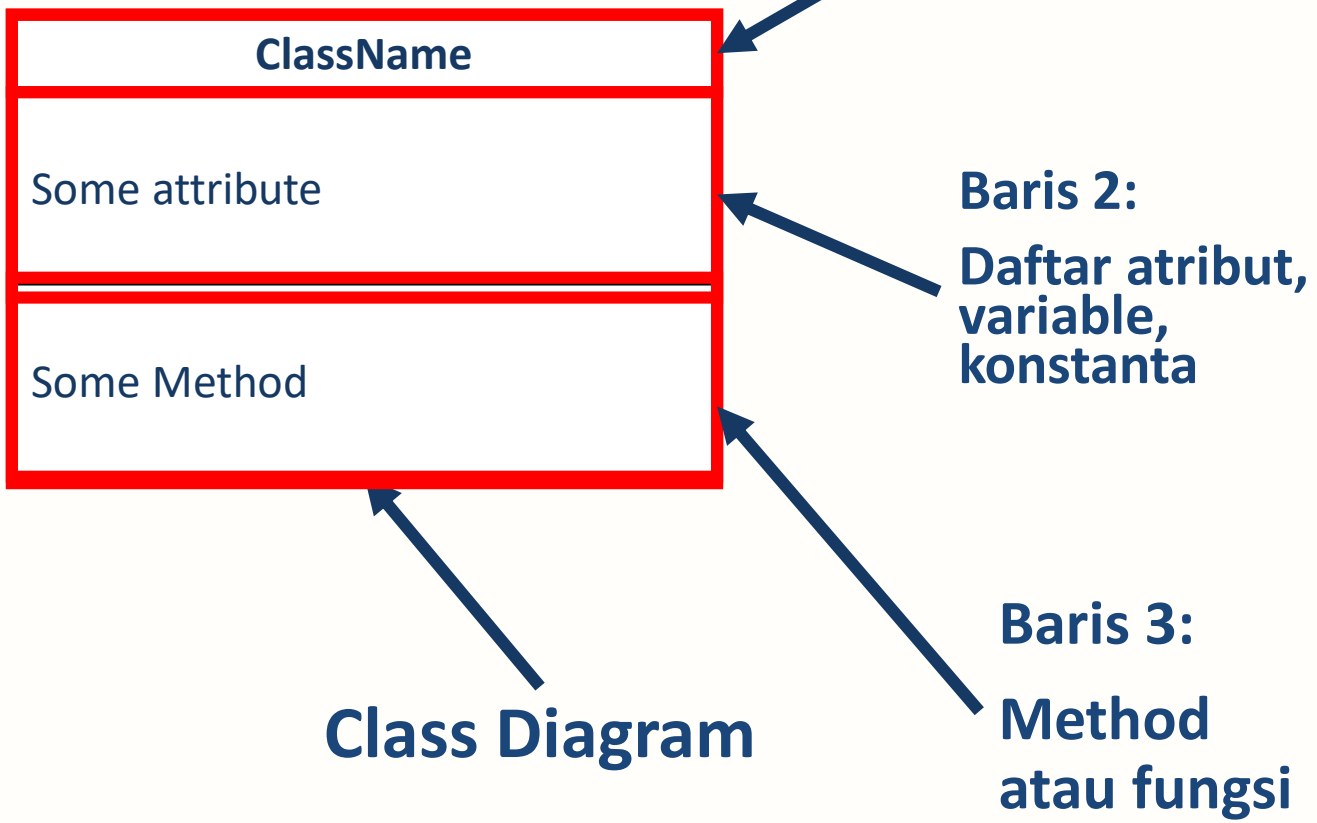
  - Public : method main() dapat diakses oleh apa saja, termasuk java technology interpreter.
  - Static : keyword ini berfungsi untuk memberi tahu kompiler bahwa method main bisa langsung digunakan dalam contex class yang bersangkutan. Untuk mengeksekusi/menjalankan method yang bertipe static, tidak diperlukan 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] …`

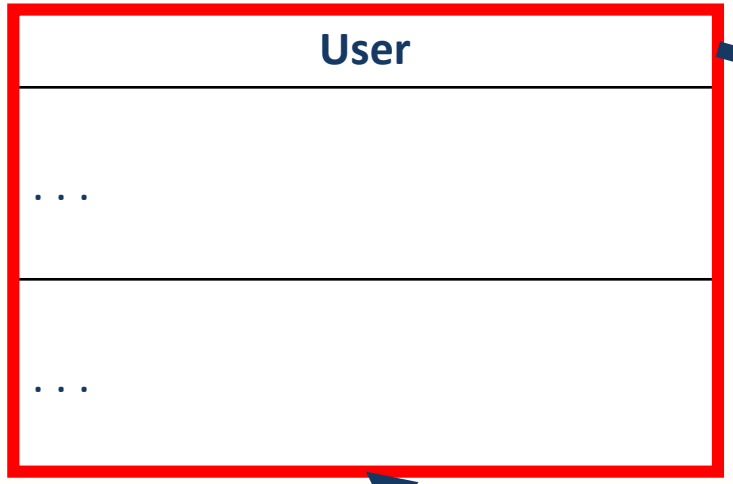
Class Diagram

Apa itu **Class Diagram**?

Diagram yang menggambarkan **Class**



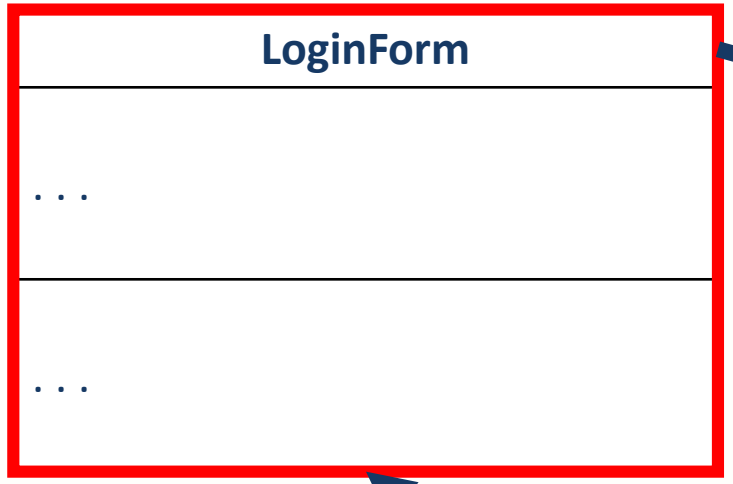
CLASS NAME



Jika di implementasikan dalam Bahasa java:

```
public class User{
    . . . .
}
```

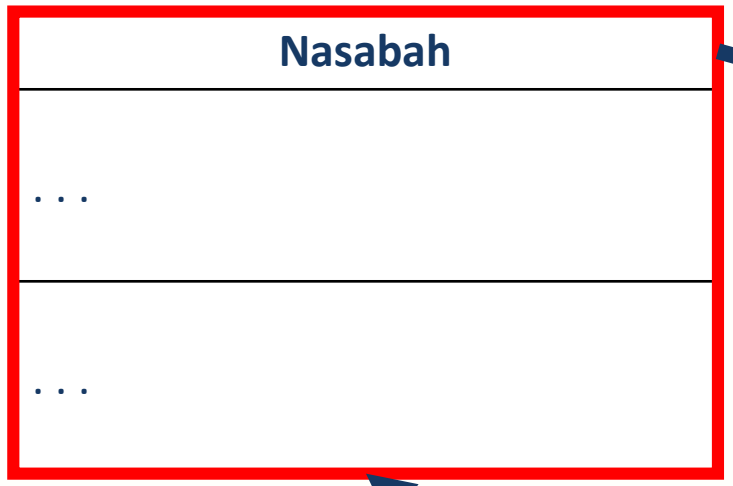
Ini adalah satu buah class dengan nama class : **User**



Jika di implementasikan dalam Bahasa java:

```
public class LoginForm{
    . . . .
}
```

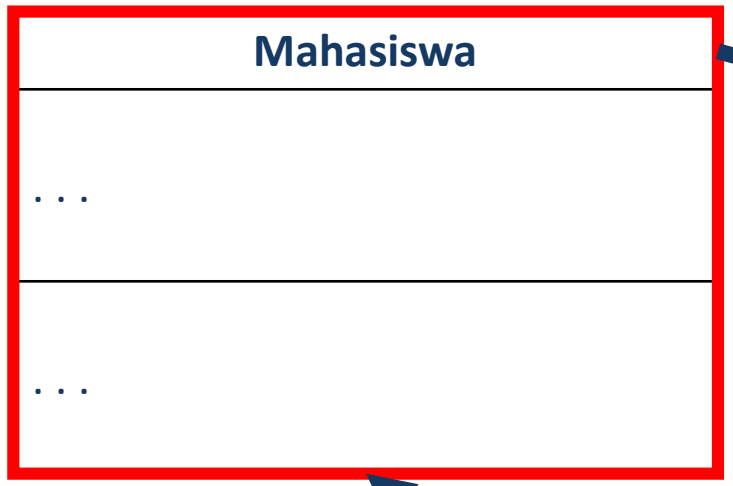
Ini adalah satu buah class dengan nama class : **LoginForm**



Jika di implementasikan dalam Bahasa java:

```
public class Nasabah{
    . . . .
}
```

Ini adalah satu buah class dengan nama class : **Nasabah**

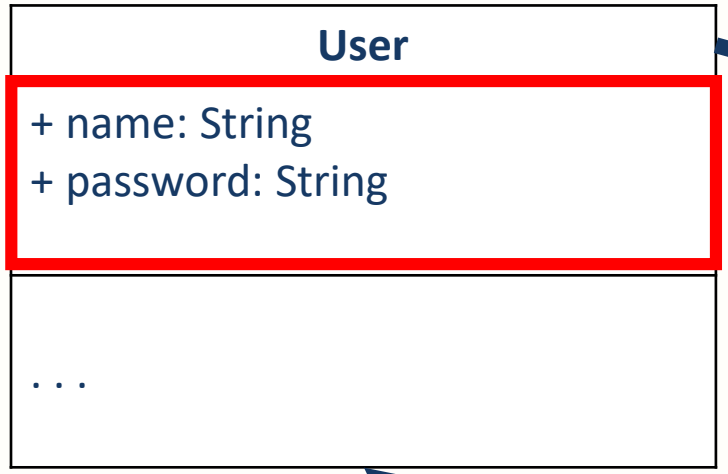


Jika di implementasikan dalam Bahasa java:

. . . ?

Ini adalah satu buah class dengan nama class : **Mahasiswa**

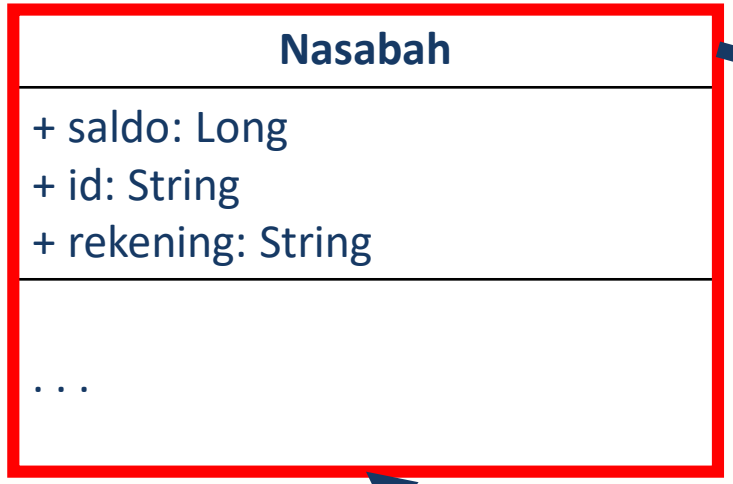
ATTRIBUTE



Dalam class User terdapat dua variable : name dan password

Jika di implementasikan dalam Bahasa java:

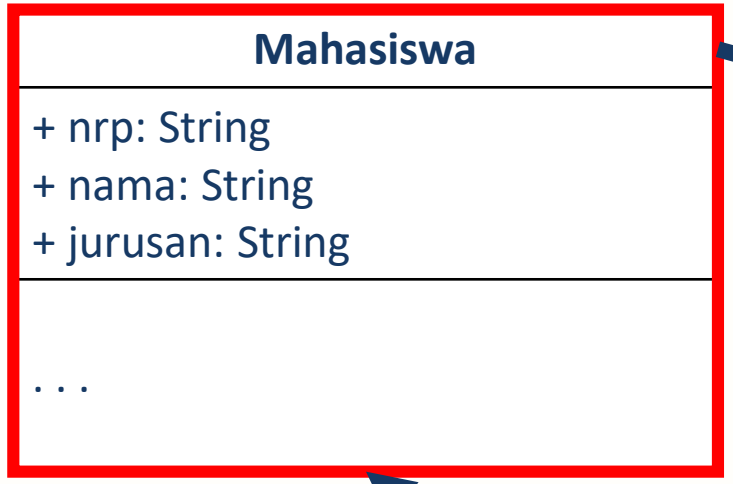
```
public class User{
    public String name;
    public String password;
}
```



Dalam class Nasabah terdapat 3 buah variable: saldo, id, rekening

Jika di implementasikan dalam Bahasa java:

```
public class Nasabah{
    public Long saldo;
    public String id;
    public String rekening;
}
```

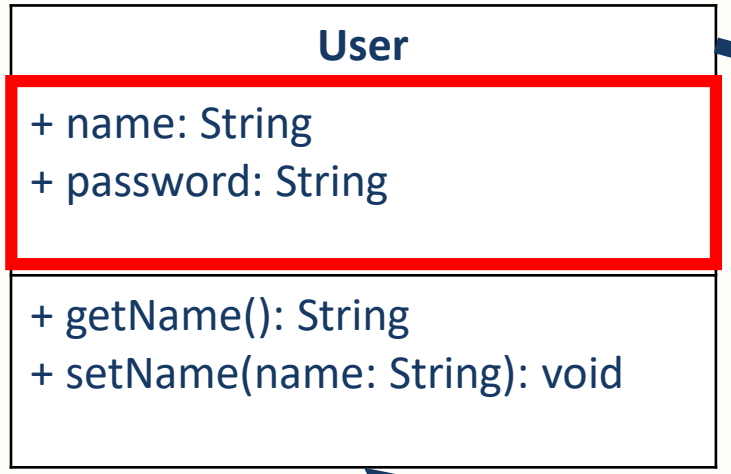


Jika di implementasikan dalam Bahasa java:

. . . ?

Dalam class Mahasiswa ada 3 variable: nrp, nama, jurusan

METHOD

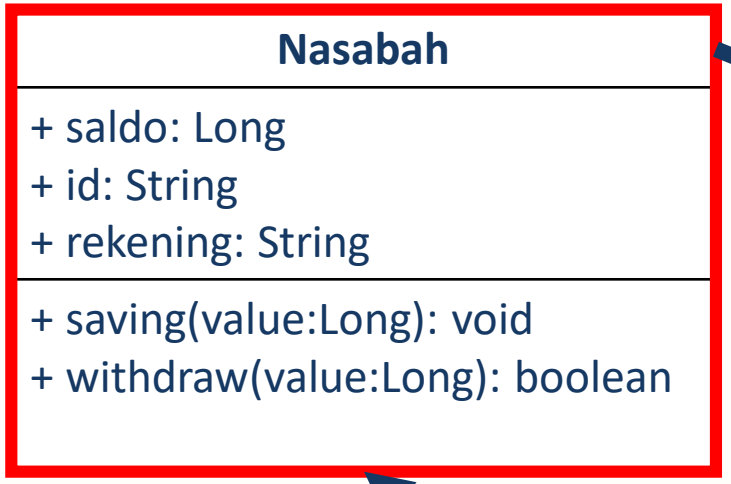


Dalam class User terdapat dua method : getName() dan setName()

Jika di implementasikan dalam Bahasa java:

```
public class User{
    public String name;
    public String password;

    public String getName(){…}
    public void setName(String name){…}
}
```

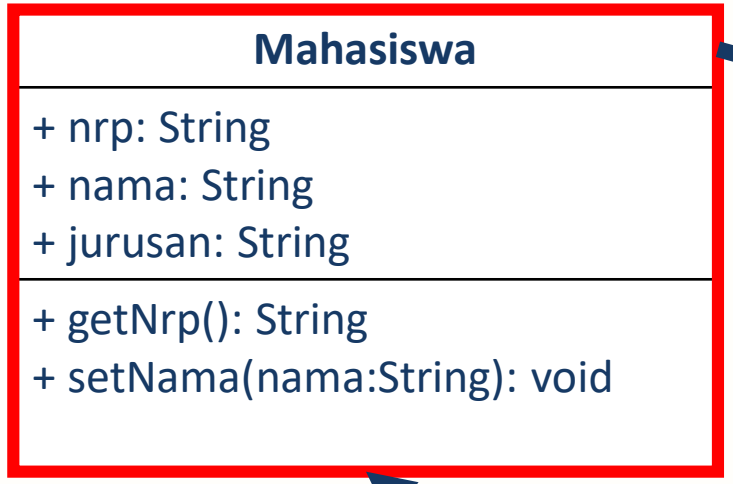


Jika di implementasikan dalam Bahasa java:

```
public class Nasabah{
    public Long saldo;
    public String id;
    public String rekening;

    public void saving(Long value){…}
    public Boolean withdraw(Long value){…}
}
```

Dalam class Nasabah terdapat 2 buah method: saving() dan withdraw()



Jika di implementasikan dalam Bahasa java:

. . . ?

Dalam class Mahasiswa ada 2 method: getNrp() dan setNama()

MODIFIER

+	adalah **public**
-	adalah **private**
#	adalah **protected**
~	adalah **default**

Mahasiswa
+ nrp: String - nama: String # jurusan: String ~ wali: String
. . .

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

bridge to the future

http://www.eepis-its.edu