Chapter 6
Konsep OOD with Java

A. Overview

- Analisis dan Desain Berorientasi Objek dapat didesain menggunakan UML namun pada implementasinya dibutuhkan bahasa pemrograman yang mendukung aplikasi objek seperti JAVA
- Pada chapter ini akan dibahas bagaimana membangun ilustrasi pada studi kasus Bank menggunakan Java
- Aplikasi dimulai dari pembuatan kelas account selanjutnya relasi antar kelas dan aplikasinya pada bank secara umum

B. Ilustration

- Ilustrasi yang dibangun pada kasus bank sebagai berikut :

  Akan dikembangkan sistem komputer yang dibutuhkan oleh suatu Bank. Bank tersebut ingin mengenalkan program layanan baru untuk komunitas lokal dan tidak sama dengan bank nasional. Sistem tersebut memungkinkan seorang nasabah untuk membuka rekening dan melakukan transaksi biasa (contoh: kredit, tabungan,) bank juga menyediakan data total aset yang dimiliki pada pemerintah

C. Penerapan UML

- Dari ilustrasi sebelumnya terdapat relasi one to many antara class Bank dan class Account. Sebagai contoh diagram sebagai berikut :

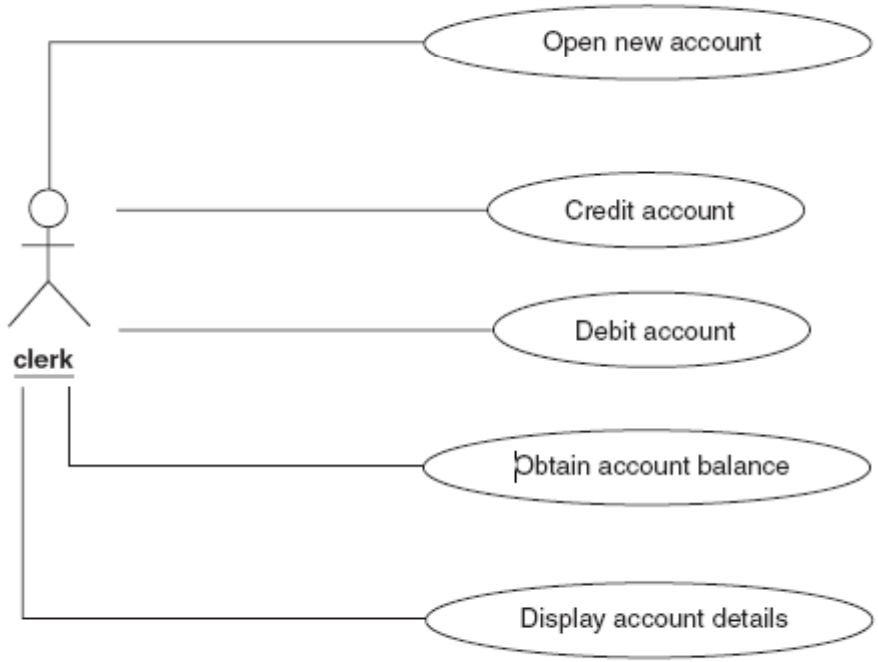

Figure 3.1 *Account use-cases*

Penerapan UML.....

- Kasus pembukaan account baru

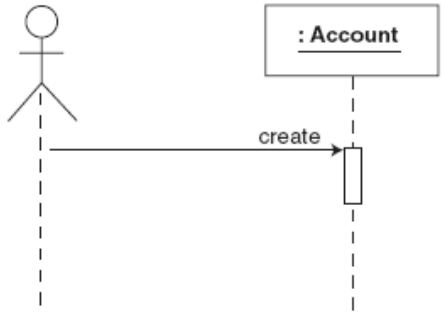

Figure 3.2 *Sequence diagram realization for "Open new account" use-case*

- Kasus pembukaan kredit

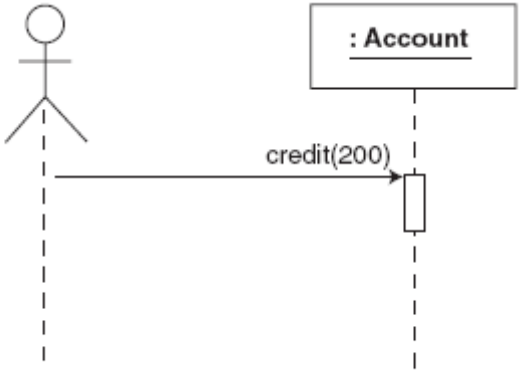

Figure 3.3 *Sequence diagram realization for "Credit account" use-case*

Penerapan UML....

- Transaksi Tabungan

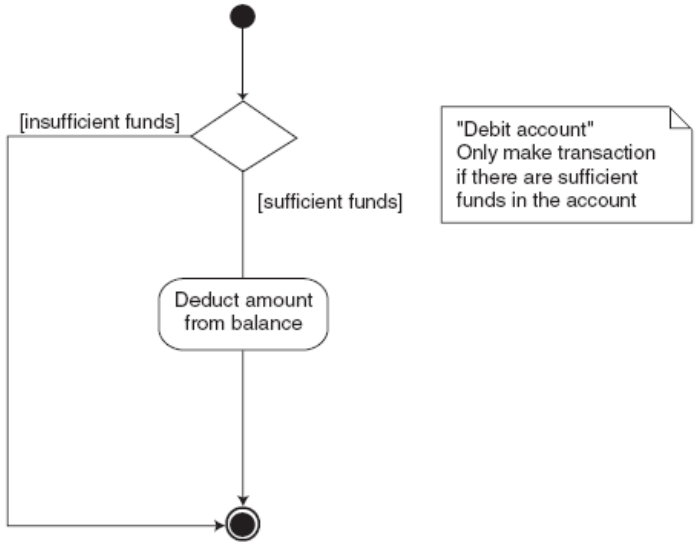

Figure 3.4 *Activity diagram for "Debit account"*

- Account Detail

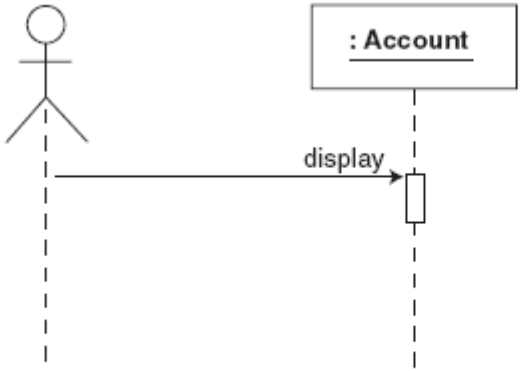

Figure 3.5 *Sequence diagram realization for "Display account details" use-case*

Penerapan UML.....

- Diagram Kelas Account

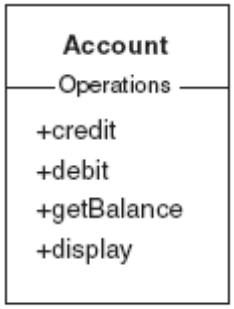

Figure 3.6(a) *Account class diagram*

- Deklarasi Kelas Account pada Java

```
public class Account{
// ----- Operations ----------
public ... credit( ... ) { ... }
public ... debit( ... ) { ... }
public ... getBalance( ... ) { ... }
public ... display( ... ) { ... }
}
```

D. Penerapan pada JAVA

- Pengembangan dari kelas account sebelumnya

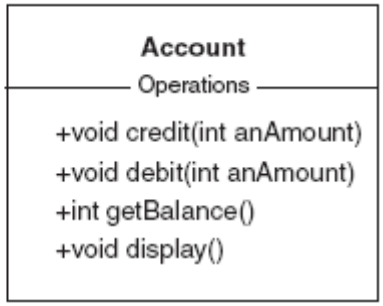

Figure 3.6(b) *Augmented Account class*

- Maka deklarasi kelas Account menjadi:

```
public class Account{
// ----- Operations ----------
public void credit(int anAmount) { ... }
public void debit(int anAmount) { ... }
public int getBalance() { ... }
public void display() { ... }
}
```

Penerapan pada Java....

- Kelas diagram dengan atribut menjadi

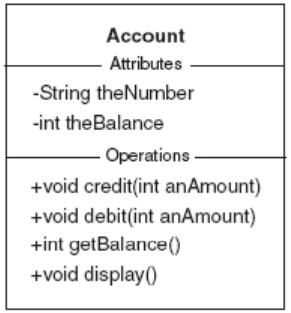

Figure 3.6(c) *Account class with attributes*

- Sehingga deklarasi pada Java menjadi :

```
public class Account{
// ----- Operations ----------
public void credit(int anAmount) { ... }
public void debit(int anAmount) { ... }
public int getBalance() { ... }
public void display() { ... }
// ----- Attributes ----------
private String theNumber;
private int theBalance;
}
```

Deklarasi lengkap pada Java

```
import textio.*;
public class Account {
// ----- Operations ----------
public void credit(int anAmount) {
theBalance += anAmount;
} // method: credit
public void debit(int anAmount) {
if(theBalance >= anAmount)
theBalance -= anAmount;
} // method: debit
public int getBalance() {
return theBalance;
} // method: getBalance
public void display() {
ConsoleIO.out.println("Account");
ConsoleIO.out.println("\t" + "Number: " + theNumber);
ConsoleIO.out.println("\t" + "Balance: " + theBalance);
} // method: display
// ----- Attributes ----------
private String theNumber;
private int theBalance;
} // class: Account
```

E. Pembangunan Aplikasi

- Seperti pada chapter sebelumnya telah dijelaskan bahwa Java mengeksekusi perintah main(), oleh karena itu dari class-class sebelumnya diimplementasikan maka kerangka sintak programnya menjadi :

```
public class Main{
// ----- Operations ----------
public static void main(String[ ] args) {
Application app = new Application();
app.run();
} // method: main
} // class: Main
```

Pembangunana aplikasi

- Diagram aplikasi objek

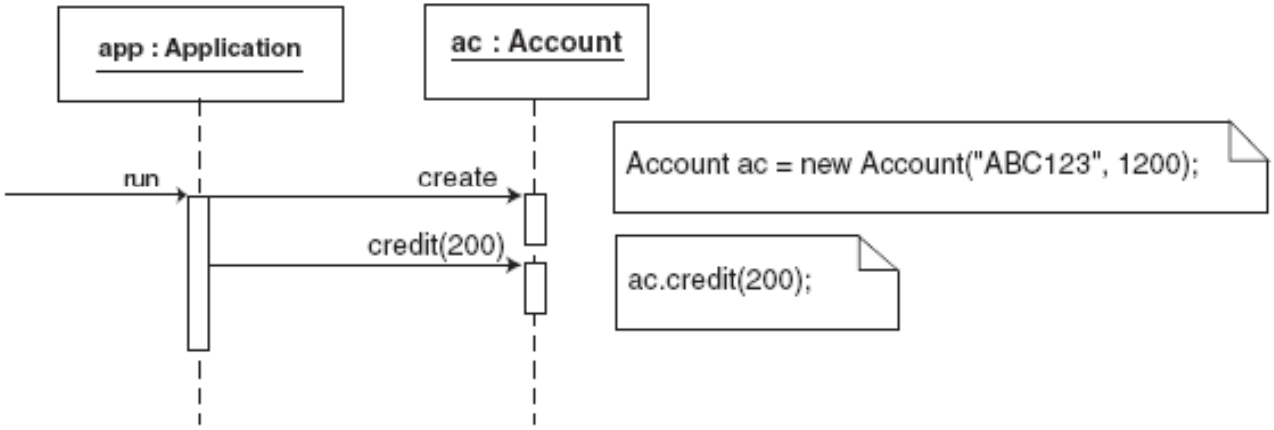

Figure 3.9 *Application object*

Pembangunan Aplikasi

- Penerapan diagram pada sintak Java

```
public class Application{
public void run(){
//
// Use-case: Open new account
//
Account ac = new Account(“ABC123”, 1200);
//
// Use-case: Credit account
//
ac.credit(200); // balance now 1400
//
// Use-case: Display account details
//
ac.display();
//
// Use-case: Others
//
ac.debit(900); // balance now 500
ac.debit(700); // balance unchanged
ac.display();
} // method: run
} // class: Application
```

Pembangunan aplikasi....

- Hasil output dari program slide sebelumnya adalah :

```
Account
Number: ABC123
Balance: 1400
Account
Number: ABC123
Balance: 500
```

F. Penerapan hubungan antar objek

- Berdasarkan studi kasus Bank tersebut memiliki kebutuhan sebagai berikut :
  - Create a bank object
  - Open a new account
  - Perform a transaction on an account
  - Obtain the total assets of the bank

Penerapan hubungan antar objek

- Diagram hubungan antara Bank dan Nasabah

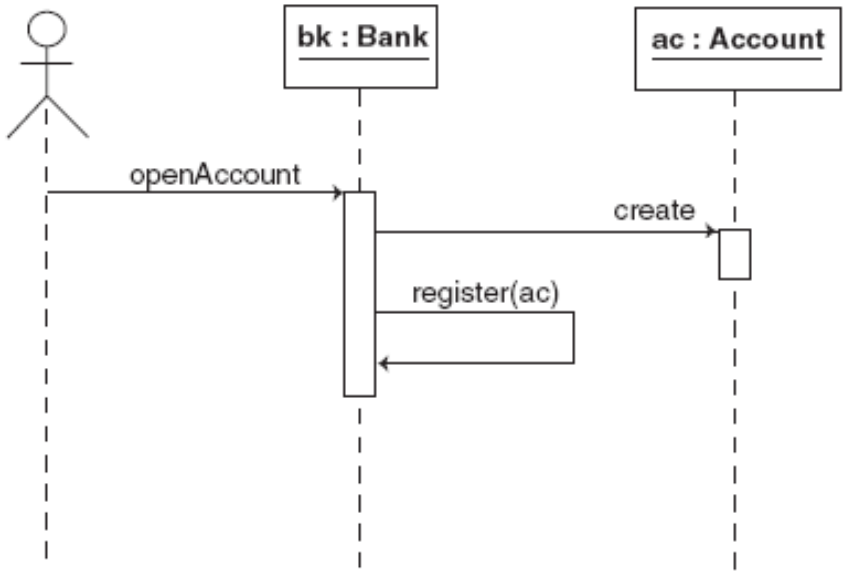

Figure 3.10(a) *Realization of the use-case "Open new account"*

Penerapan hubungan antar objek

- Dari diagram tersebut diaplikasikan pada pembuatan objek sebagai berikut

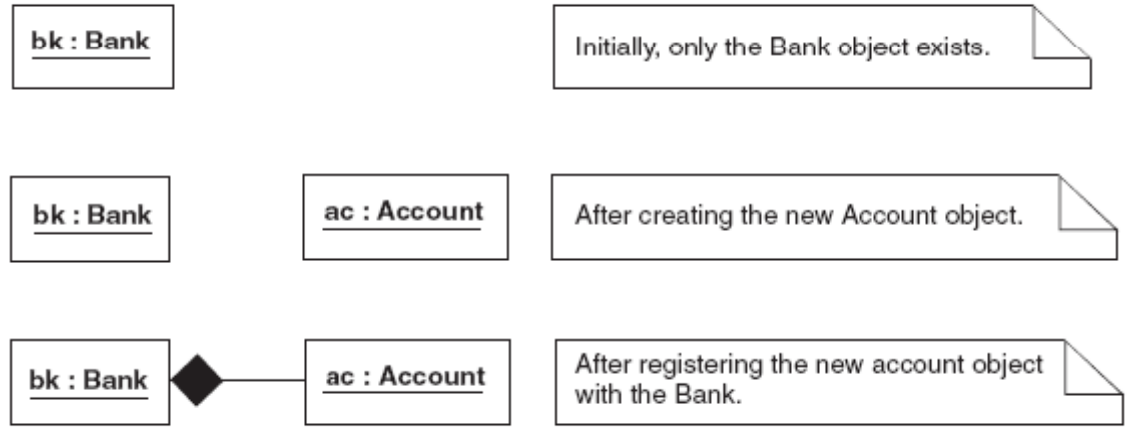

Figure 3.10(b) *Account object aggregated into the Bank object*

Penerapan hubungan antar objek

- Diagram hubungan Bank dan Nasabah pada kasus kredit

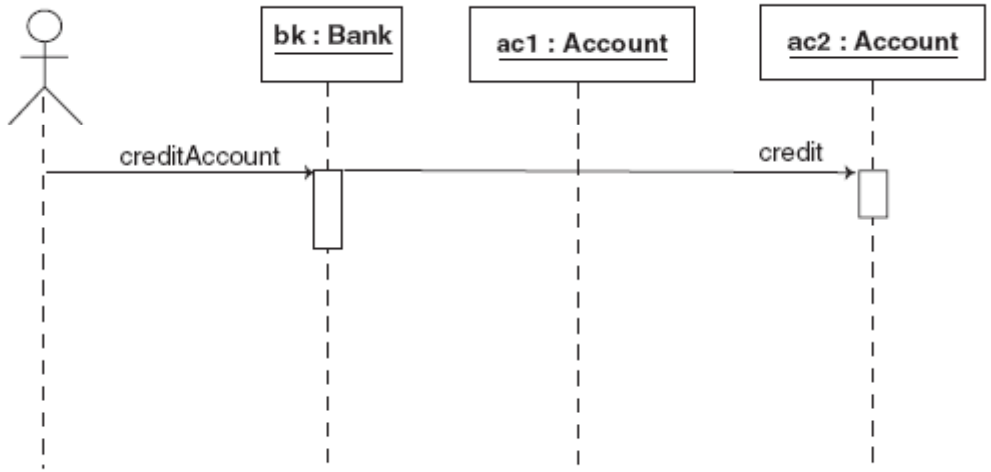

Figure 3.11 *The use-case "Perform a transaction (credit) on an account"*

Penerapan hubungan antar objek

- Diagram hubungan Bank dan Nasabah pada kasus penghitungan total aset

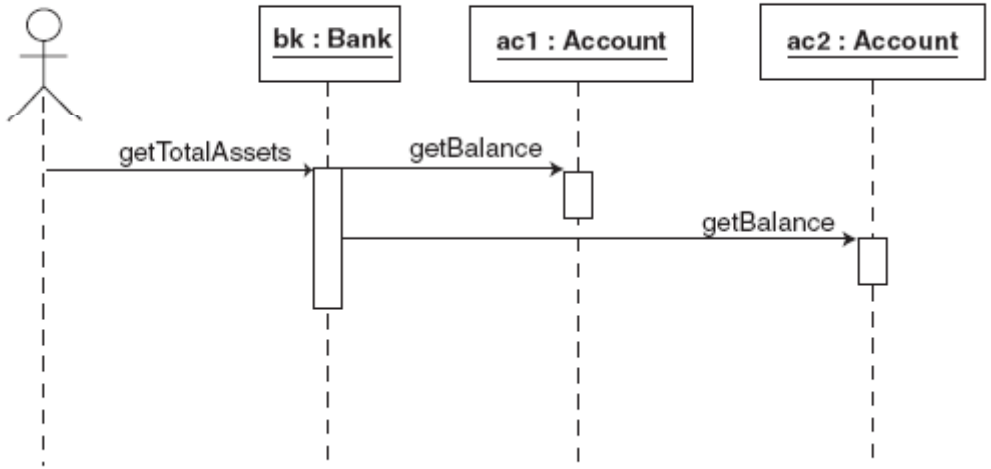

Figure 3.12 *Realizing the use-case "Obtain the total assets of the bank"*

G. Penerapan program

1. Pembuatan class Bank

```
public class Bank{
public void openAccount(String aNumber, int anInitialBalance) {
Account acc = new Account(aNumber, anInitialBalance);
theAccounts.add(acc);
} // method: openAccount
// ...
// ----- Relations ----------
private java.util.ArrayList theAccounts; // of Account
} // class: Bank
```

Penerapan program.....

2. Pembuatan class Account

```
public class Account implements java.lang.Comparable{
// ----- Operations ----------
public Account(String aNumber, int anInitialBalance) { ... }
public Account() { ... }
public boolean equals(Object obj) { ... }
public int compareTo(Object obj) { ... }
// ...
} // class: Account
```

DAFTAR PUSTAKA

- Barclay K., Savage J., 2004, *E-book Object Oriented Design with UML and Java*