

Pemrograman I

Literatur

- H.M Deitel, P.J Deitel, **Small Java How to Program**-sixth Edition, Pearson Prentice Hall, 2005
- Elliot B. Koffman, Paul A.T. Wolfgang, **Objects, Abstraction, Data Structures and Design Using Java**, John Wiley & Sons.Inc, 2005
- Ian F. Darwin, **Java Cookbook**, O'Reilly, 2001
- Mark Allen Weiss, **Data Structures & Algorithm Analysis in Java**, Addison-Wesley, 1999
- Moh.Sjukani, **Algoritma & Struktur Data dengan C, C++ dan Java**, Mitra Wacana Media, Agustus 2005
- Rangsang Purnama, **Tuntunan Pemrograman Java jilid- 1**, Prestasi Pustaka Publisher, Januari 2003
- Rangsang Purnama, **Tuntunan Pemrograman Java jilid- 2**, Prestasi Pustaka Publisher, Juli 2003
- Rangsang Purnama, **Tuntunan Pemrograman Java jilid 3**, Prestasi Pustaka Publisher, Maret 2003
- Ariesto Hadi Sutopo, Fajar Masya, **Pemrograman Berorientasi Objek dengan Java**, Graha Ilmu, 2005
- Indrajani, Martin, **Pemrograman Berorientasi Objek dengan Java**, Elex Media Komputindo, 2004
- Melvin Antonius, Damian Bayu Imam Santoso, Carneles, **Membuat Animasi dengan Java**, Elex Media Komputindo, 2004

Materi ***praktikum**

1. **Pengertian Java**
2. **Setup /Instalasi Java**
3. **Version Control (Git)**
4. **Anatomi aplikasi Java**
5. **Classpath**
6. **Variabel dan Tipe Data**
7. **Operator**
8. **Control Flow (If.... Else, For/While)**
9. **Class & Object**
10. **Method**
11. **Exception**
12. **Konsep OOP**
13. **Inheritance**
14. **Encapsulation**
15. **Polymorphism**
16. **Abstract Class & Interface**
17. **Composition & Aggregation**
18. **Studi Kasus & Presentasi**

METHOD & EXCEPTION

Method & Exception

- Deklarasi Method

Boolean	ApakahAktif	()
Return type	Nama	Argumen

public	static	void	main	(String[] Xx)	throws Exception
Access modifier	static	Return type	nama	argument	Tipe data exception

private	Integer	tambah	(Integer X,	Integer Y)
Access modifier	return	nama	Argument1	Argument2

- Apabila return type datanya integer maka argumen 1 dan argumen 2 harus integer juga
- Void → artinya tidka mengeluarkan hasil

Exception Handling

- Exception suatu hal yang terjadi dalam aplikasi yang tidak kita harapkan atau biasa juga disebut error
- Isi dari exception handling adalah
 1. Jenis Exception
 2. Ada object exception itu sendiri. Dapat menimbulkan exception. Object Exception atau Raise Exception
 3. Handle
 - a. Bisa ditangkap → catch
 - b. Bisa dilempar → throw

Exception Handling

- Contoh → Terdapat daftar nama orang, kita akan baca dalam bentuk text file
- Langkahnya adalah
 1. Buka File
 - Kemungkinan errornya, yaitu :
 - Tidak dapat dibuka filenya
 - Filenya tidak ada
 2. Loop, baca file dan tampilkan
 - Kemungkinan errornya, yaitu :
 - Tidak dapat dibaca filenya
 - Filenya tidak ada
 3. Tutup file
 - Kemungkinan errornya, yaitu :
 - Gagal (akan dilakukan proses tutup, file sudah kondisi terhapus)

Exception Handling

- Programmer yang baik biasanya dilihat dari exception handling yang digunakannya.
- Semakin banyak exception handling yang dipakai semakin profesional yang dibuat program

Exception Handling

- Contoh
 - Mekanisme koding yang bukan object oriented:

```
Void prosesfile() {  
    int status =0;  
    status =file open("halo.txt")  
    if(status == 0) {  
        data file baca();  
        while(status == 0) {  
            print(data.isi)  
        }  
    }  
    Status = file close()  
}
```

- Status code diatas:
 1. Susah mengingat kode errornya dibaris berapa
 2. Campur antara businness logic dengan error handling
 3. Tidak ada perpanjangan propagation
 4. Tidak ada keterangan error

Exception Flow menggunakan

```
try {
```

```
    File open();
```

```
    while(data = File.baca() {  
        tampilkan(data);
```

```
    }
```

```
    File close();
```

```
}
```

```
catch(FileNotFoundException err) { //tidak bisa baca file  
    tampilkan(err)
```

```
}
```

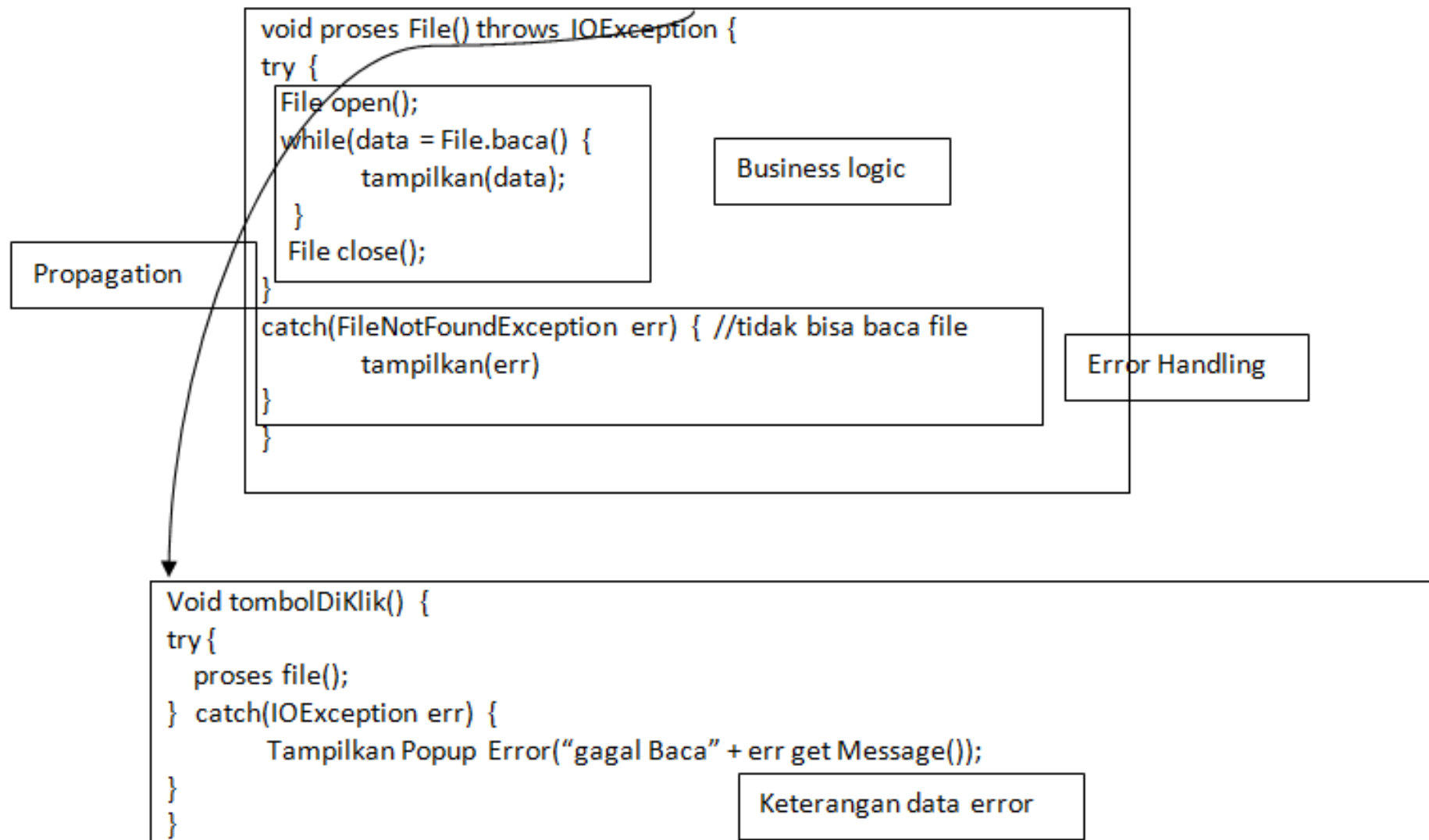
```
catch(IOException err) { //tidak bisa membaca input dan output  
    tampilkan(err)
```

```
}
```

Business logic

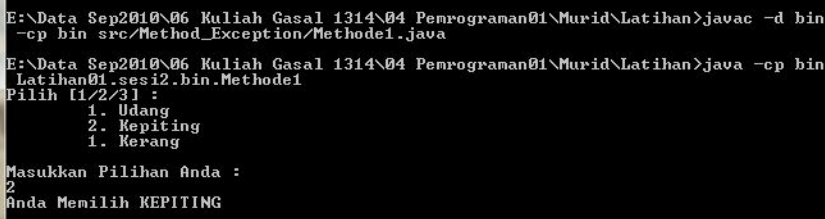
Error Handling

Exception Flow menggunakan



Contoh 1: Method & Exception

```
import java.io.*;
public class Method1 {
    static BufferedReader input = new BufferedReader(new InputStreamReader(System.in));
    public static void main(String[] Xx) {
        try {
            System.out.println("Pilih [1/2/3] : " + "\n\t1. Udang" + "\n\t2. Kepiting" + "\n\t1. Kerang");
            System.out.println();
            System.out.println("Masukkan Pilihan Anda : ");
            String choose = input.readLine();
            if(choose.equals("1")) {
                System.out.println("Anda Memilih UDANG");
            }
            else
            if(choose.equals("2")) {
                System.out.println("Anda Memilih KEPITING");
            }
            else
            if(choose.equals("3")) {
                System.out.println("Anda Memilih KERANG");
            }
            else {
                throw new Exception("Pilihan Anda Tidak Tersedia !!!");
            }
        }
        catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```



```
E:\Data Sep2010\06 Kuliah Gasal 1314\04 Pemrograman01\Murid\Latihan>javac -d bin
-cp bin src/Method_Exception/Method1.java
E:\Data Sep2010\06 Kuliah Gasal 1314\04 Pemrograman01\Murid\Latihan>java -cp bin
Latihan01.sesi2.bin.Method1
Pilih [1/2/3] :
1. Udang
2. Kepiting
1. Kerang

Masukkan Pilihan Anda :
2
Anda Memilih KEPITING
```

KONSEP

OOP (Object Oriented Programming)

Cara Pendekatan

- Pendekatan/cara pandang dapat dilihat dari :
 1. OOP
 2. Prosedural
 3. Functional

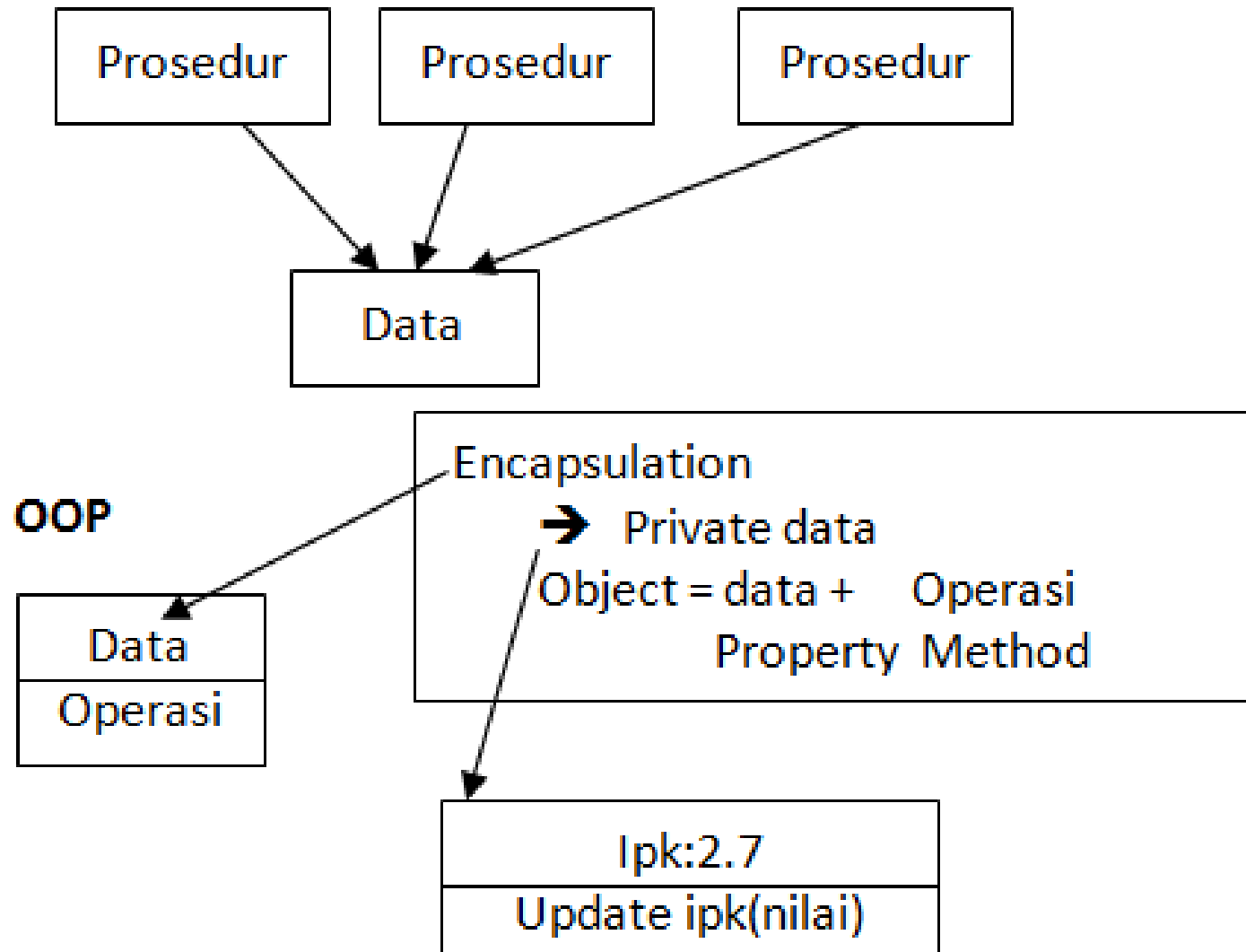
1. Object Oriented Programming

- a. Encapsulation
- b. Polymorphism → ada diskon di looping dari jenis diskon → Dapat memilih implementasi apa yang dijalankan pada saat proses
→ method apa yang digunakan pada saat object dijalankan → biasanya dikatakan runtime binding
- c. Inheritance
 - Tipe data → super class -- umum, sub class – khusus → terdiri dari class dan interface
 - Inherited → properti/methods selain itu dapat juga menjadi

1. Object Oriented Programming

- Method yang tidak ada implementasinya disebut dengan method abstract
- Class isinya harus diimplementasi
- Abstract method mengharuskan mempunyai abstract class
- Apabila semuanya abstract dapat ditulis menjadi interfase
- Interface adalah suatu deklarasi data yang abstract semua
- Interface untuk membuat suatu hal

2. Prosedural



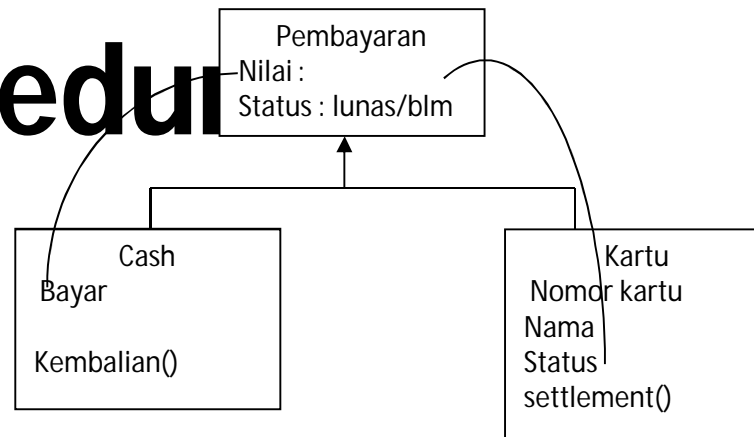
2. Prosedural

- Inheritance → turunan (kalau untuk program dengan kata **is a** atau **adalah**)
- Instance adalah object yang dibuat
- contoh object mahasiswa mempunyai instance rangga, amir, susi
- Contoh Transaksi penjualan di INDOMART

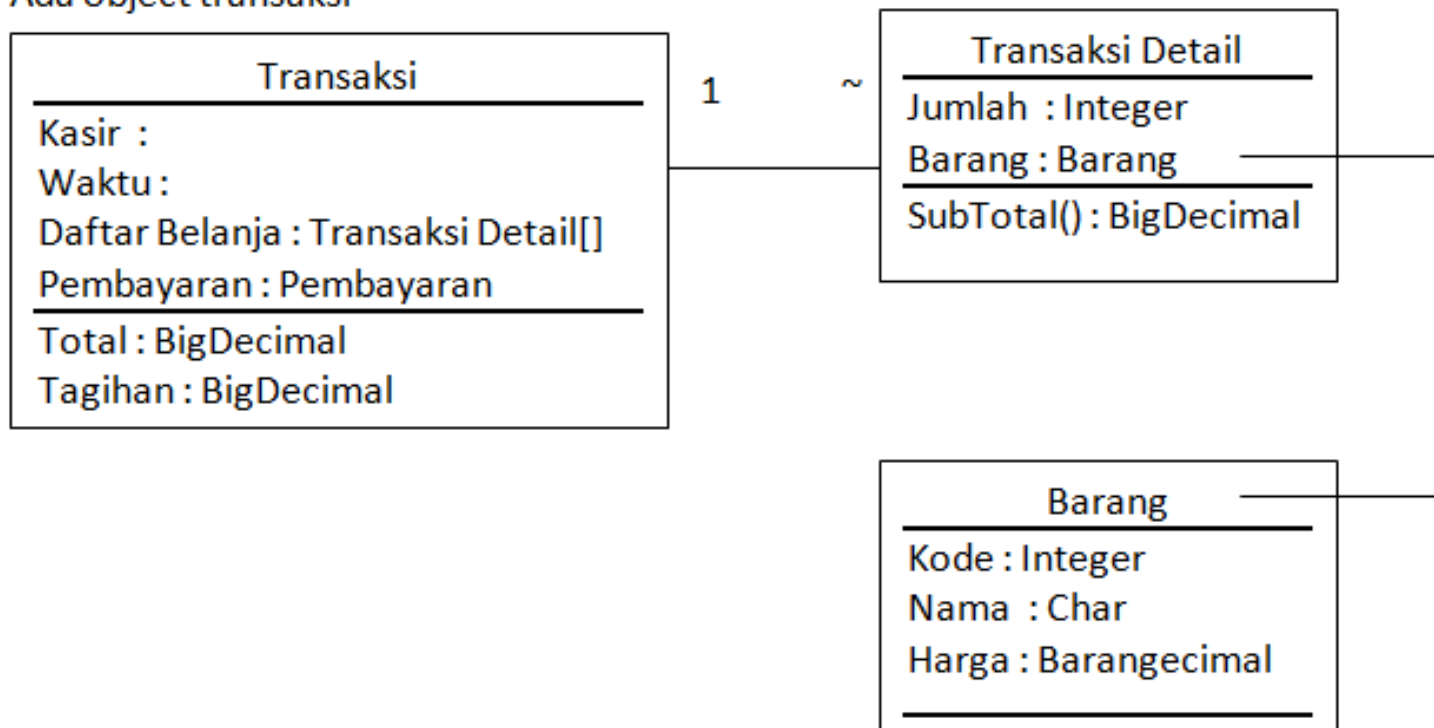
Kasir :			
Waktu :			
Kode	Qrty	Harga	Sub Total
Susu	1	100	100
Kopi	2	300	600
			Total : 700
Bayar: 1000			
Kembali : 300			
Bayar			

Inheritance
Is a
adalah

2. Prosedur

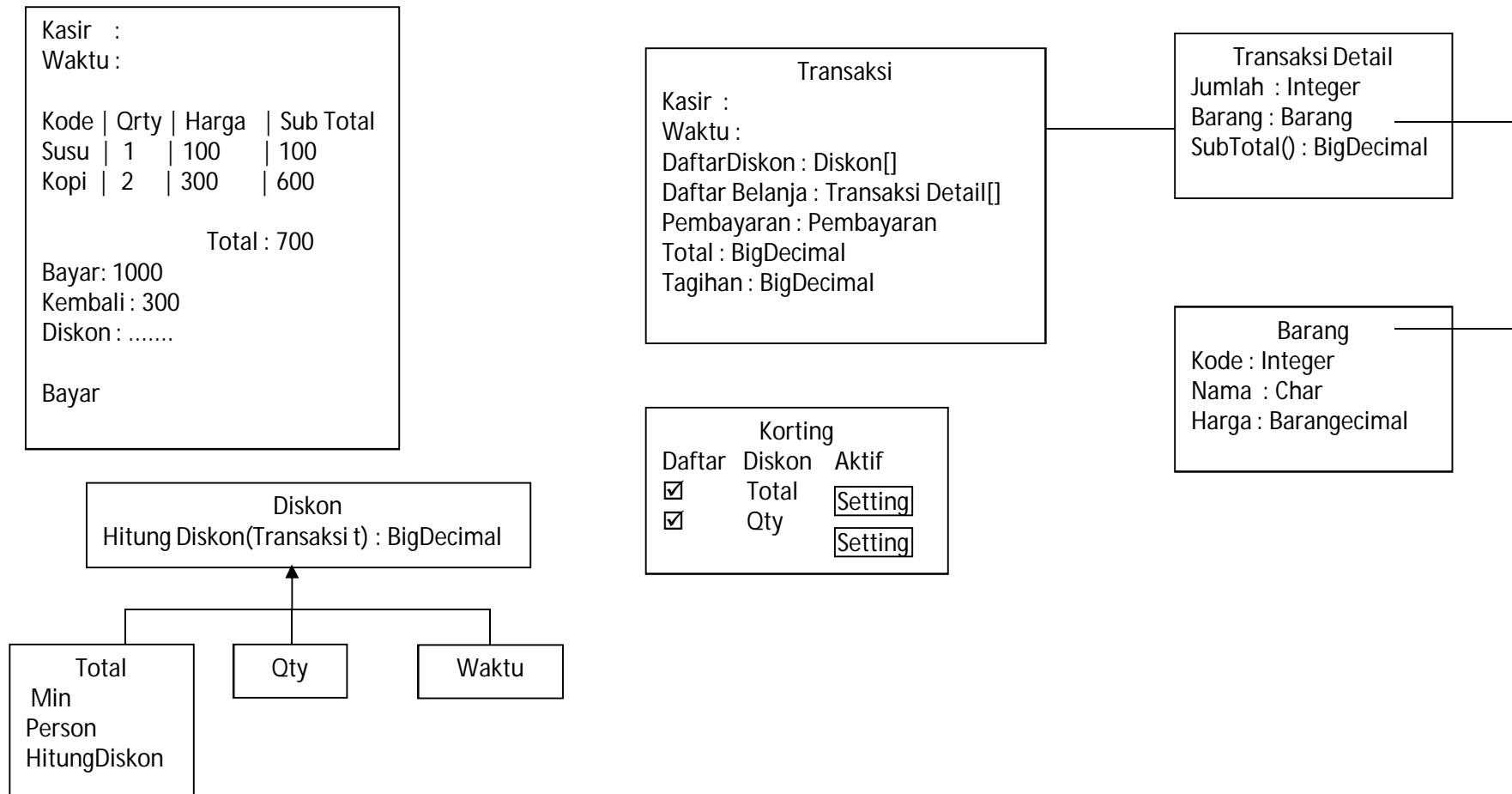


Ada object transaksi



2. Prosedural

- Apabila menggunakan diskon, maka akan menjadi:



Studi Kasus

Class : Barang

```
//Perintah Object Oriented Programming
package Latihan01.sesi3.bin;
import java.math.BigDecimal;

public class Barang      {
    private String kode;
    private String nama;
    private BigDecimal harga;

    public String getKode()    {
        return kode;
    }
    public void setKode(String x)  {
        kode = x;
    }
    public String getNama()    {
        return nama;
    }
    public void setNama(String x)  {
        nama = x;
    }
    public BigDecimal getHarga()  {
        return harga;
    }
    public void setHarga(BigDecimal x)  {
        harga = x;
    }
}
```

Studi Kasus

Class : Pembayaran

```
//Perintah Object Oriented Programming
package Latihan01.sesi3.bin;
import java.math.BigDecimal;

public class Pembayaran {
    private BigDecimal nilai;

    public BigDecimal getNilai() {
        return nilai;
    }

    public void setNilai(BigDecimal x) {
        nilai = x;
    }
}
```

Studi Kasus

Class :
TransaksiDetail

```
//Perintah Object Oriented Programming
package Latihan01.sesi3.bin;
import java.math.BigDecimal;

public class TransaksiDetail {
    private Barang barang;
    private Integer jumlah;

    public BigDecimal subtotal() {
        return barang.getHarga().multiply(new BigDecimal(jumlah));
    }
}
```

Studi Kasus

Class : Kartu

```
//Perintah Object Oriented Programming
package Latihan01.sesi3.bin;
import java.math.BigDecimal;

public class Kartu extends Pembayaran {
    private String nomer;
    private String nama;

    public String getNomor() {
        return nomer;
    }

    public void setNomor(String x) {
        nomer = x;
    }

    public void settlement() {
        //ToDo : buat implementasinya
    }
}
```


Studi Kasus

Class :
DiskonPeriod
e

```
//Perintah Object Oriented Programming
package Latihan01.sesi3.bin;
import java.math.BigDecimal;

public class DiskonPeriode implements Diskon {
    public BigDecimal hitung(Transaksi t) {
        //to do :implement perhitungan yang benar
        return BigDecimal.ZERO;
    }
}
```

Studi Kasus

Class :
DiskonJumlah

```
//Perintah Object Oriented Programming
package Latihan01.sesi3.bin;
import java.math.BigDecimal;

public class DiskonJumlah implements Diskon {
    private BigDecimal persentase;
    private BigDecimal nilaiMinimum;

    public BigDecimal hitung(Transaksi t) {
        if(t.hitungTagihan().compareTo(nilaiMinimum) > 0) {
            return persentase.multiply(t.hitungTagihan());
        }
        else {
            return BigDecimal.ZERO;
        }
    }
}
```

Studi Kasus

Class : Cash

```
//Perintah Object Oriented Programming
package Latihan01.sesi3.bin;
import java.math.BigDecimal;

public class Cash extends Pembayaran {
    private BigDecimal dibayar;
    private Transaksi transaksi;

    public BigDecimal getDibayar() {
        return dibayar;
    }

    public void setDibayar(BigDecimal x) {
        dibayar = x;
    }

    public BigDecimal hitungKembalian() {
        return transaksi.hitungTagihan().subtract(dibayar);
    }
}
```

```

package Latihan01.sesi3.bin;
import java.math.BigDecimal;
import java.util.Date;
import java.util.List;
import java.util.ArrayList;

public class Transaksi {
    private String kasir;
    private Date waktu;

    private List<TransaksiDetail> daftarBelanja = new ArrayList<TransaksiDetail>();
    private List<Diskon> daftarDiskon = new ArrayList<Diskon>();
    private Pembayaran pembayaran;

    public BigDecimal hitungTotal() {
        BigDecimal total = BigDecimal.ZERO;

        for(TransaksiDetail td : daftarBelanja) {
            total = total.add(td.subtotal());
        }
        return total;
    }

    public BigDecimal hitungTagihan() {
        BigDecimal totalDiskon = BigDecimal.ZERO;
        for(Diskon d:daftarDiskon) {
            totalDiskon = totalDiskon.add(d.hitung(this));
        }
        return hitungTotal().subtract(totalDiskon);
    }
}

```

Studi Kasus

Class
:
Transaksi

Studi Kasus

Class : Diskon

```
//Perintah Object Oriented Programming
package Latihan01.sesi3.bin;
import java.math.BigDecimal;

public interface Diskon {
    public BigDecimal hitung(Transaksi t);
}
```