JOBSHEET II  
OBJECT

1. Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

1. Mengenal objek dan class sebagai konsep mendasar pada pemrograman berorientasi objek
2. Mendeklarasikan class, atribut dan method
3. Membuat objek (instansiasi)
4. Mengakses atribut dan method dari suatu objek
5. Menerapkan konstruktor
6. Praktikum
7. Percobaan 1: Deklarasi Class, Atribut dan Method

**Waktu Percobaan : 50 Menit**

Pada Percobaan 1 ini dilakukan pembuatan class beserta atribut dan method yang dimilikinya. Perhatikan Class Diagram berikut ini:

|  |
| --- |
| Buku |
| judul: String  pengarang: String  halaman: int  stok: int  harga: int |
| tampilInformasi(): void  terjual(jml: int): void  restock(n: int): void  gantiHarga(hrg: int): int |

Berdasarkan class diagram tersebut, akan dibuat program mengguanakan bahasa Java.

1. Langkah-langkah Percobaan
2. Buka text editor. Buat file baru, beri nama **Buku<NoAbsen>.java**
3. Lengkapi class **Buku** dengan atribut yang telah digambarkan di dalam class diagram tersebut

A group of black and blue letters

Description automatically generated

1. Lengkapi class **Buku** dengan method yang telah digambarkan di dalam class diagram tersebut



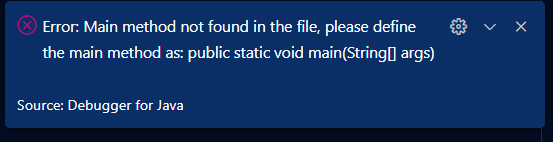


1. Compile dan run program.
2. Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program Anda dengan gambar berikut ini.

A screenshot of a computer

Description automatically generated



1. Pertanyaan
2. Sebutkan dua karakteristik class atau object!

**Jawab:** encapsulation, inheritance

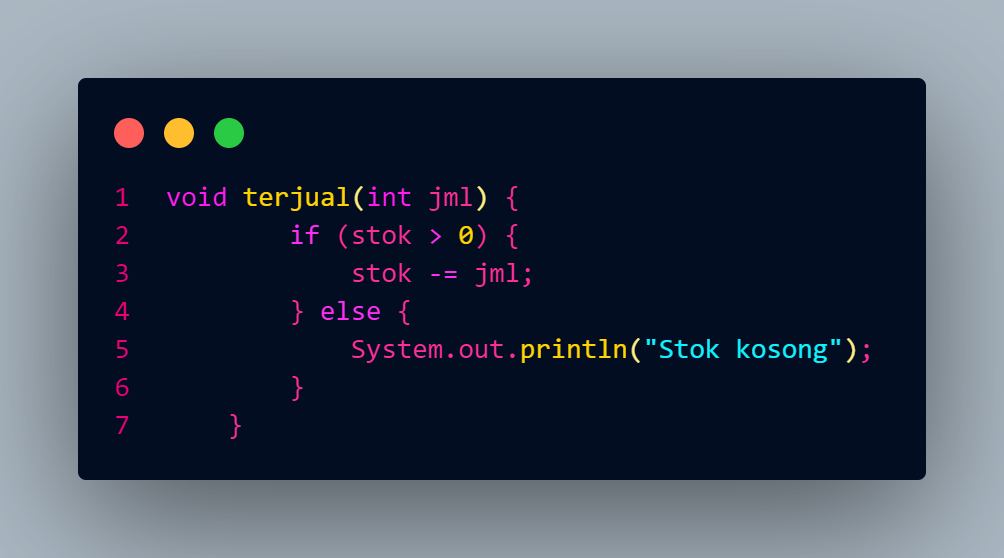
1. Perhatikan class **Buku** pada Praktikum 1 tersebut, ada berapa atribut yang dimiliki oleh class Buku? Sebutkan apa saja atributnya!

**Jawab:** ada 5 atribut yaitu, judul;pengarang;jumlah halaman;sisa stok;harga

1. Ada berapa method yang dimiliki oleh class tersebut? Sebutkan apa saja methodnya!

**Jawab:** ada 4 method yaitu, tampilInformasi;terjual;restock;gantiHarga

1. Perhatikan method **terjual()** yang terdapat di dalam class **Buku**. Modifikasi isi method tersebut sehingga proses pengurangan hanya dapat dilakukan jika stok masih ada (lebih besar dari 0)!

**Jawab: **

1. Menurut Anda, mengapa method **restock()** mempunyai satu parameter berupa bilangan int?

**Jawab:** karena yang dibutuhkan hanya int saja dan membuat method mudah digunakan, fleksibel, aman, dan efisien

1. **Commit dan push kode program ke Github**

**Jawab:**

1. Percobaan 2: Instansiasi Object, serta Mengakses Atribut dan Method

**Waktu Percobaan: 50 Menit**

Sampai tahap ini, class **Buku** telah berhasil dibuat pada Percobaan 1. Selanjutnya, apabila class Buku tersebut ingin digunakan dan diakses atribut serta method-nya, maka perlu dibuat object/instance dari class **Buku** terlebih dahulu melalui proses instansiasi.

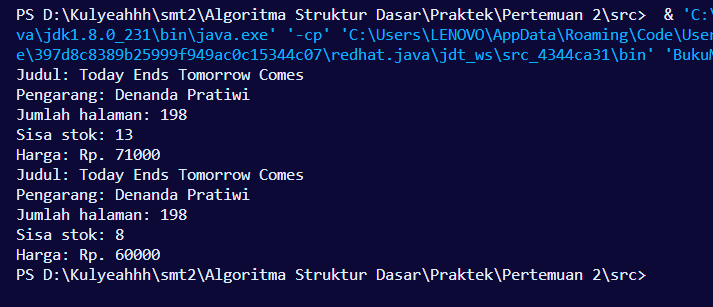
* + 1. Langkah-langkah Percobaan

1. Buat file baru, beri nama **BukuMain<NoAbsen>.java**
2. Tuliskan struktur dasar bahasa pemrograman Java yang terdiri dari fungsi **main()**
3. Di dalam fungsi **main(),** lakukan instansiasi, kemudian lanjutkan dengan mengakses atribut dan method dari objek yang telah terbentuk.

A screenshot of a computer code

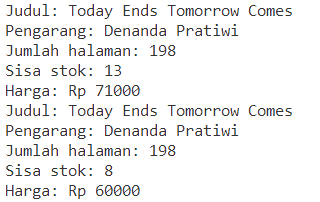
Description automatically generated

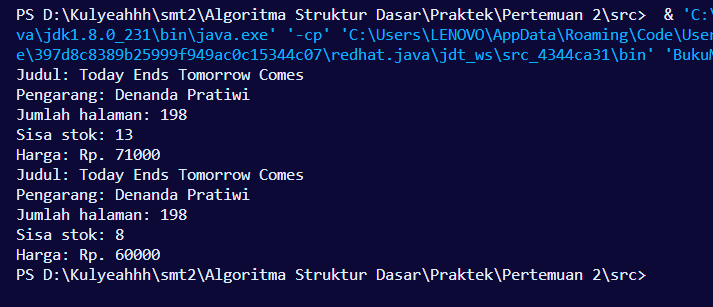
1. Compile dan run program.



1. **Commit dan push kode program ke Github**
   * 1. Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program Anda dengan gambar berikut ini.





* + 1. Pertanyaan

1. Pada class **BukuMain**, tunjukkan baris kode program yang digunakan untuk proses instansiasi! Apa nama object yang dihasilkan?

**Jawab:** nama object yang dihasilkan merupakan bk1



1. Bagaimana cara mengakses atribut dan method dari suatu objek?

**Jawab:** dengan menggunakan titik (.)

1. Mengapa hasil output pemanggilan method **tampilInformasi()** pertama dan kedua berbeda?

**Jawab:** karena **tampilInformasi()** yang pertama digunakan untuk menampilkan data yang belum mengalami perubahan, sedangkan **tampilInformasi()** yang kedua digunakan untuk menampilkan data setelah mengalami perubahan.

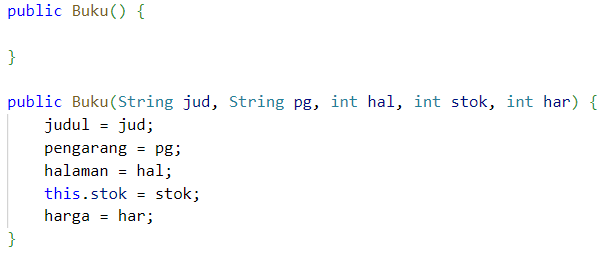
1. Percobaan 3: Membuat Konstruktor

**Waktu Percobaan: 60 Menit**

Pada percobaan ini, dilakukan pembuatan kode program untuk mengimplementasikan berbagai macam konstruktor berdasarkan parameternya.

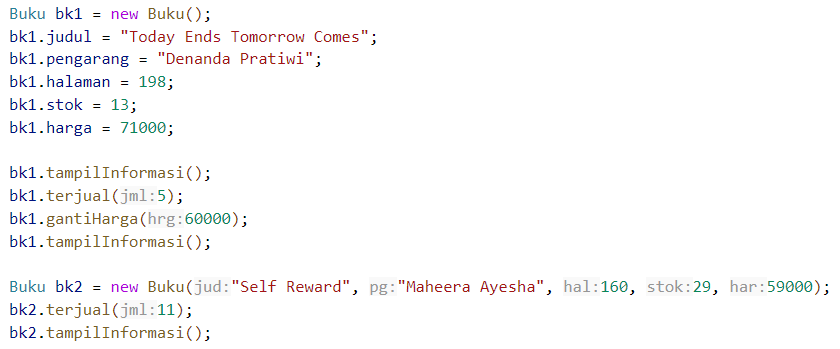
* + 1. Langkah-langkah Percobaan

1. Buka kembali class **Buku**. Tambahkan dua buah konstruktor di dalam class **Buku** tersebut, yang terdiri dari satu konstruktor default dan satu konstruktor berparameter. Konstruktor merupakan method Istimewa, penempatan kode program untuk konstruktor dapat diperlakukan sama seperti method yang lain (setelah atribut).



*Catatan: Apabila nama parameter sama dengan nama atribut, maka untuk merujuk pada variabel atribut ditambahkan sintaks* ***this*** *di depan nama* ***atribut***

1. Buka kembali class **BukuMain**. Buat sebuah object lagi bernama **bk2** dengan menggunakan konstruktor berparameter.



1. Compile dan run program.
2. **Commit dan push kode program ke Github**
   * 1. Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program Anda dengan gambar berikut ini.

A screenshot of a computer

Description automatically generated

* + 1. Pertanyaan

1. Pada class **Buku** di Percobaan 3, tunjukkan baris kode program yang digunakan untuk mendeklarasikan konstruktor berparameter!
2. Perhatikan class **BukuMain**. Apa sebenarnya yang dilakukan pada baris program berikut?



1. Hapus konstruktor default pada class **Buku**, kemudian compile dan run program. Bagaimana hasilnya? Jelaskan mengapa hasilnya demikian!
2. Setelah melakukan instansiasi object, apakah method di dalam class **Buku** harus diakses secara berurutan? Jelaskan alasannya!
3. Buat object baru dengan nama **buku<NamaMahasiswa>** menggunakan konstruktor berparameter dari class **Buku**!
4. **Commit dan push kode program ke Github**
   1. Latihan Praktikum

**Waktu : 150 Menit**

1. Pada class Buku yang telah dibuat, tambahkan tiga method yaitu **hitungHargaTotal()**, **hitungDiskon()**, dan **hitungHargaBayar()** dengan penjelasan sebagai berikut:
   * Method **hitungHargaTotal()** digunakan untuk menghitung harga total yang merupakan perkalian antara harga dengan jumlah buku yang terjual
   * Method **hitungDiskon()** digunakan untuk menghitung diskon dengan aturan berikut:
     + Jika harga total lebih dari 150000, maka harga didiskon sebesar 12%
     + Jika harga total antara 75000 sampai 150000, maka harga didiskon sebesar 5%
     + Jika harga total kurang dari 75000, maka harga tidak didiskon
   * Method **hitungHargaBayar()** digunakan untuk menghitung harga total setelah dikurangi diskon

Class diagram **Buku** setelah penambahan ketiga method tersebut adalah sebagai berikut.

|  |
| --- |
| Buku |
| judul: String  pengarang: String  halaman: int  stok: int  harga: int |
| tampilInformasi(): void  terjual(jml: int): void  restock(n: int): void  gantiHarga(hrg: int): int  **hitungHargaTotal(): int**  **hitungDiskon(): int**  **hitungHargaBayar(): int** |

1. Buat program berdasarkan class diagram berikut ini!

|  |
| --- |
| Dragon |
| x: int  y: int  width: int  height: int |
| moveLeft(): void  moveRight(): void  moveUp(): void  moveDown(): void  printPosition(): void  detectCollision(x: int, y: int): void |

Penjelasan dari atribut dan method pada class Dragon tersebut adalah sebagai berikut:

* + Atribut **x** digunakan untuk menyimpan posisi koordinat x (mendatar) dari dragon, sedangkan atribut **y** untuk posisi koordinat y (vertikal)
  + Atribut **width** digunakan untuk menyimpan lebar dari area permainan, sedangkan **height** untuk menyimpan panjang area
  + Method **moveLeft()** digunakan untuk mengubah posisi dragon ke kiri (koordinat x akan berkurang 1), sedangkan **moveRight()** untuk bergerak ke kanan (koordinat x akan bertambah 1). Perlu diperhatikan bahwa koordinat x tidak boleh lebih kecil dari 0 atau lebih besar dari nilai width. Jika koordinat x < 0 atau x > width maka panggil method **detectCollision()**
  + Method moveUp() digunakan untuk mengubah posisi dragon ke atas (koordinat y akan berkurang 1), sedangkan moveDown() untuk bergerak ke bawah (koordinat y akan bertambah 1). Perlu diperhatikan bahwa koordinat y tidak boleh lebih kecil dari 0 atau lebih besar dari nilai height. Jika koordinat y < 0 atau y > height maka panggil method **detectCollision()**
  + Method **detectCollision()** akan mencetak pesan “Game Over” apabila dragon menyentuh ujung area permainan.