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

**/\*\***

**\* Buku12**

**\*/**

public class Buku12 {

    String judul, pengarang;

    int halaman, stok, harga;

    void tamilInformasi(){

        System.out.println("judul : " + judul);

        System.out.println("pengarang : " + pengarang);

        System.out.println("jumlah halaman : " + halaman);

        System.out.println("sisa stok : " + stok );

        System.out.println("harga : Rp " + harga);

    }

    void terjual (int jml ){

        stok -= jml;

    }

    void restok (int jml){

        stok += jml;

    }

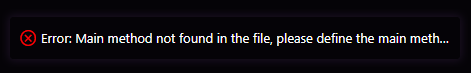
    void gantiHarga (int jml){

        harga = hrg ;

    }

}

**Output :**

****

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

**1. Encapsulation: Menggabungkan data dan metode dalam satu unit, menyembunyikan detail implementasi, dan hanya mengekspos metode yang relevan.**

**2. Inheritance: Memungkinkan class untuk mewarisi sifat-sifat (data dan metode) dari class lain, memungkinkan untuk mendefinisikan ulang, menambahkan, atau memodifikasi perilaku class yang sudah ada.**

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

**Class Buku pada Praktikum 1 memiliki lima atribut, yaitu:**

* **judul (String)**
* **pengarang (String)**
* **halaman (int)**
* **stok (int)**
* **harga (int)**

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

**Class Buku pada Praktikum 1 memiliki 4 method :**

* **tampilInformasi()**
* **terjual(int jml)**
* **restok(int jml)**
* **gantiHarga(int jml)**

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)!

 void terjual (int jml ){

       if ( stok > 0){

            stok -= jml;

    } else {

        System.out.println("Stok habis, tidak dapat melakukan penjualan.");

    }

}

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

**Method `restock()` menggunakan satu parameter berupa bilangan bulat (`int`) karena stok buku umumnya dihitung dalam bilangan bulat, sehingga tipe data `int` dipilih untuk menentukan jumlah tambahan stok yang akan ditambahkan.**

1. **Commit dan push kode program ke Github**
2. 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

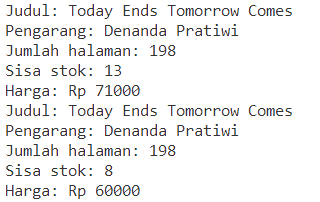
1. Buat file baru, beri nama **BukuMain<NoAbsen>.java**
2. Tuliskan struktur dasar bahasa pemrograman Java yang terdiri dari fungsi **main()**
3. Di dalam fungsi **main(),** lakukan instansiasi, kemudian lanjutkan dengan mengakses atribut dan method dari objek yang telah terbentuk.

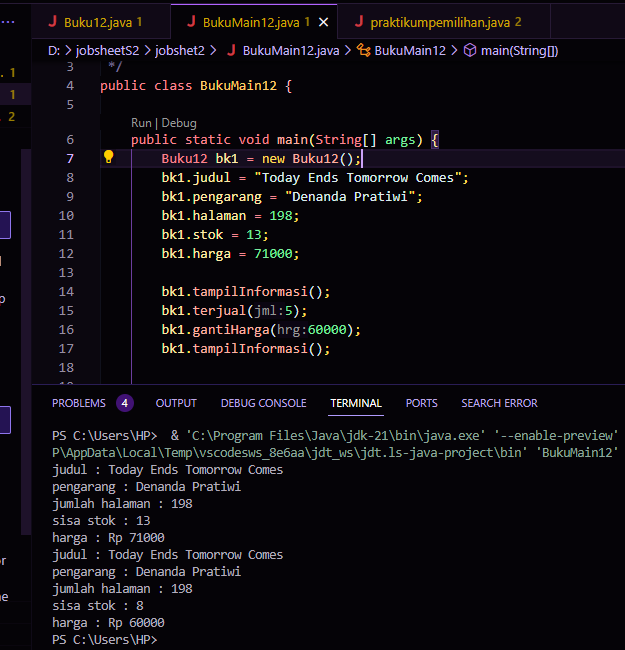
A screenshot of a computer code

Description automatically generated

1. Compile dan run program.
2. **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?

**Buku12 bk1 = new Buku12();**

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

**Untuk mengakses atribut dan method dari suatu objek, menggunakan nama objek yang telah dibuat, diikuti oleh tanda titik (.), kemudian diikuti dengan nama atribut atau method yang ingin diakses.**

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

**Perbedaan output terjadi karena nilai `stok` dan `harga` objek `bk1` telah diubah antara pemanggilan pertama dan kedua method `tampilInformasi()`. Pada pemanggilan pertama, `stok` adalah 13 dan `harga` adalah 71000. Setelah pemanggilan `terjual(5)` dan `gantiHarga(60000)`, nilai `stok` menjadi 8 dan `harga` menjadi 60000, sehingga output kedua mencerminkan perubahan ini.**

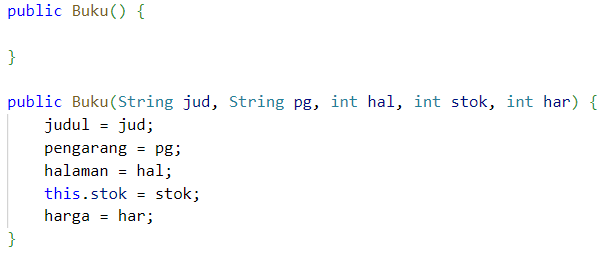
1. Percobaan 3: Membuat Konstruktor

**Waktu Percobaan: 60 Menit**

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

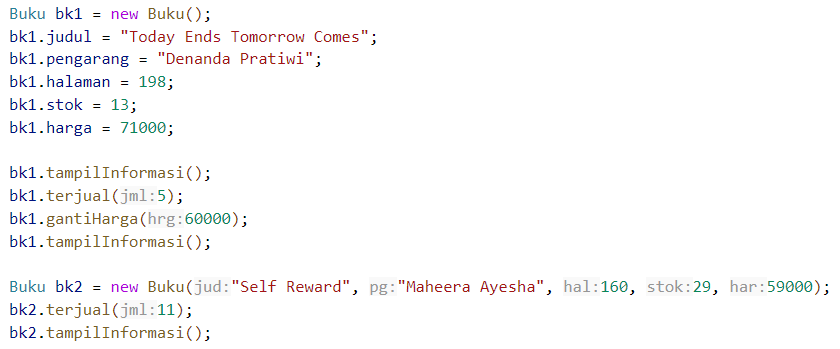
* + 1. Langkah-langkah Percobaan

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).

zz

*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.

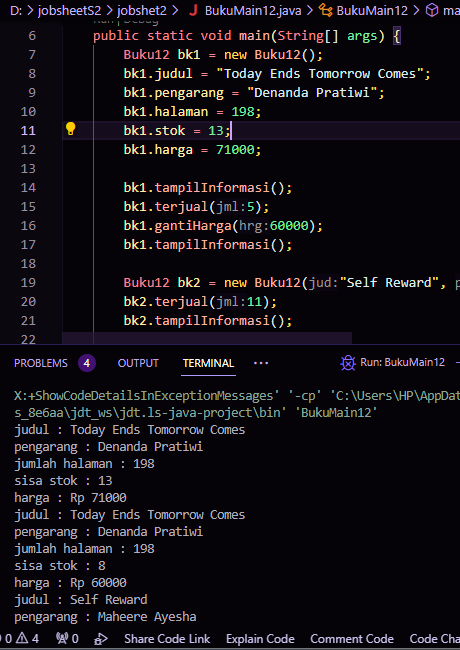


1. Compile dan run program.
2. **Commit dan push kode program ke Github**
   * 1. Verifikasi Hasil Percobaan

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!

baris kode program yang digunakan untuk mendeklarasikan konstruktor berparameter yaitu ada 5 :

* jud (judul)
* pg (pengarang)
* hal (halaman)
* stok (stok)
* har (harga).

1. Perhatikan class **BukuMain**. Apa sebenarnya yang dilakukan pada baris program berikut?



**dilakukan pembuatan objek baru dari kelas Buku dengan menggunakan konstruktor berparameter. Objek baru tersebut diinisialisasi dengan nilai-nilai yang diberikan sebagai argumen pada konstruktor**

1. Hapus konstruktor default pada class **Buku**, kemudian compile dan run program. Bagaimana hasilnya? Jelaskan mengapa hasilnya demikian!

**akan terjadi error, kompilasi yang menyatakan bahwa tidak ada konstruktor yang sesuai yang dapat digunakan untuk membuat objek Buku12. Hal ini terjadi karena tidak ada konstruktor default yang bisa dipanggil setelah penghapusan konstruktor default.**

1. Setelah melakukan instansiasi object, apakah method di dalam class **Buku** harus diakses secara berurutan? Jelaskan alasannya!

**Tidak, method di dalam class Buku tidak harus diakses secara berurutan setelah melakukan instansiasi objek. dapat mengakses method di dalam class Buku sesuai dengan kebutuhan dan urutan yang diperlukan dalam program.**

1. Buat object baru dengan nama **buku<NamaMahasiswa>** menggunakan konstruktor berparameter dari class **Buku**!
2. **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** |

**/\*\***

**\* Buku012**

**\*/**

public class Buku012 {

    String judul;

    String pengarang;

    int jumlahHalaman;

    int sisaStok;

    double harga;

**// Konstruktor dengan parameter**

    public Buku012 (String judul, String pengarang, int jumlahHalaman, int sisaStok, double harga) {

*this*.judul = judul;

*this*.pengarang = pengarang;

*this*.jumlahHalaman = jumlahHalaman;

*this*.sisaStok = sisaStok;

*this*.harga = harga;

    }

    void tampilInformasi() {

        System.out.println("Judul: " + judul);

        System.out.println("Pengarang: " + pengarang);

        System.out.println("Jumlah halaman: " + jumlahHalaman);

        System.out.println("Sisa stok: " + sisaStok);

        System.out.println("Harga: Rp " + (int) harga);

        System.out.println();

    }

**// Method untuk mengupdate jumlah stok ketika buku terjual**

   public void terjual(int jml) {

    if (jml <= stok) {

        stok -= jml;

    } else {

        System.out.println("Stok tidak mencukupi");

    }

}

**// Method untuk menambah stok buku**

public void restock(int jml) {

    stok += jml;

}

**// Method untuk mengubah harga buku**

public void gantiHarga(int hrg) {

    harga = hrg;

}

**// Method untuk menghitung harga total**

public int hitungHargaTotal(int jumlahTerjual) {

    return harga \* jumlahTerjual;

}

**// Method untuk menghitung diskon**

public int hitungDiskon(int hargaTotal) {

    int diskon = 0;

    if (hargaTotal > 150000) {

        diskon = (int) (0.12 \* hargaTotal); **// diskon 12%**

    } else if (hargaTotal >= 75000 && hargaTotal <= 150000) {

        diskon = (int) (0.05 \* hargaTotal); **// diskon 5%**

    }

    return diskon;

}

**// Method untuk menghitung harga total setelah dikurangi diskon**

public int hitungHargaBayar(int hargaTotal, int diskon) {

    return hargaTotal - diskon;

}

    public static void main(String[] args) {

**// Membuat objek buku baru dengan konstruktor berparameter**

        Buku012 bukuBaru = new Buku012("CAHYA", "Pengarang Baru", 200, 50, 75000);

**// Menampilkan informasi buku baru**

        System.out.println("Informasi buku baru:");

        bukuBaru.tampilInformasi();

**// Menghitung harga total**

        int jumlahTerjual = 10;

        int hargaTotal = bukuBaru.hitungHargaTotal(jumlahTerjual);

        System.out.println("Harga Total: " + hargaTotal);

**// Menghitung diskon**

        int diskon = bukuBaru.hitungDiskon(hargaTotal);

        System.out.println("Diskon: " + diskon);

**// Menghitung harga bayar**

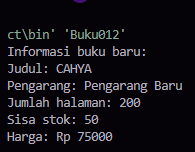
        int hargaBayar = bukuBaru.hitungHargaBayar(hargaTotal, diskon);

        System.out.println("Harga Bayar: " + hargaBayar);

    }

}

**Output :**

****

1. Buat program berdasarkan class diagram berikut ini!

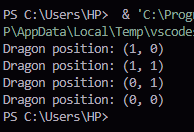
|  |
| --- |
| Dragon |
| x: int  y: int  width: int  height: int |
| moveLeft(): void  moveRight(): void  moveUp(): void  moveDown(): void  printPosition(): void  detectCollision(x: int, y: int): void |

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.

1. public class tugasp2 {
2. private int x;
3. private int y;
4. private int maxX;
5. private int maxY;
6. public tugasp2(int x, int y, int maxX, int maxY) {
7. *this*.x = x;
8. *this*.y = y;
9. *this*.maxX = maxX;
10. *this*.maxY = maxY;
11. }
12. public void moveLeft() {
13. x--;
14. if (x < 0) {
15. detectCollision();
16. x = 0; **// Resetting x**
17. }
18. }
19. public void moveRight() {
20. x++;
21. if (x > maxX) {
22. detectCollision();
23. x = maxX; **// Resetting x**
24. }
25. }
26. public void moveUp() {
27. y--;
28. if (y < 0) {
29. detectCollision();
30. y = 0;
31. }
32. }
33. public void moveDown() {
34. y++;
35. if (y > maxY) {
36. detectCollision();
37. y = maxY;
38. }
39. }
40. public void printPosition() {
41. System.out.println("Dragon position: (" + x + ", " + y + ")");
42. }
43. public void detectCollision() {
44. System.out.println("Collision Detected! Game Over!");
45. }
46. public static void main(String[] args) {
47. **// Creating an instance of TugasP2 (formerly Dragon)**
48. tugasp2 dragon = new tugasp2(0, 0, 10, 10);
49. **// Moving the dragon and printing its position**
50. dragon.moveRight();
51. dragon.printPosition();
52. dragon.moveDown();
53. dragon.printPosition();
54. dragon.moveLeft();
55. dragon.printPosition();
56. dragon.moveUp();
57. dragon.printPosition();
58. }
59. }

**Output :**

****