

LAPORAN HASIL PRAKTIKUM

ALGORITMA DAN STRUKTUR DATA

JOBSHEET 2



OLEH :

IVAN RIZAL AHMADI

NIM. 2341760128

SIB-1F/13

D-IV SISTEM INFORMASI BISNIS

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2.1.3 Pertanyaan percobaan 1

1. Sebutkan dua karakteristik class atau object! =

- Encapsulation: Class atau objek dapat menggabungkan data (atribut) dan metode (fungsi) ke dalam satu kesatuan, sehingga menyembunyikan detail implementasi dari luar dan mendorong pemisahan tanggung jawab.

-Inheritance: Class dapat mewarisi sifat dan perilaku dari class lain, memungkinkan untuk membangun hierarki class dan menghindari duplikasi kode.

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

- judul (title)

-pengarang (author)

-halaman (number of pages)

-stok (stock)

-harga (price)

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

-tampilinformasi(): Menampilkan informasi tentang buku.

-terjual(int jml): Mengurangi stok buku saat terjadi penjualan.

-restock(int jml): Menambah stok buku saat proses restocking.

-gantiHarga(int hrg): Mengubah harga buku.

4. 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;  
        System.out.println(jml + " buku terjual. Sisa stok: " + stok);  
    } else {  
        System.out.println(x:"Stok habis. Tidak dapat melakukan penjualan.")  
    }  
}
```

5. Menurut Anda, mengapa method **restock()** mempunyai satu parameter berupa bilangan int? = Parameter jml (jumlah) dalam metode restock mewakili jumlah buku yang akan ditambahkan ke stok. Dengan adanya parameter ini, metode tersebut menjadi lebih

fleksibel, memungkinkan pemanggil untuk menentukan berapa banyak buku yang harus di-restock setiap kali metode dipanggil. Parameterisasi ini memungkinkan penyesuaian dinamis dari jumlah restocking berdasarkan kebutuhan aplikasi.

6. Commit dan push kode program ke Github =

A screenshot of a code editor with a dark background and light-colored text. The code is in Java and defines a class named 'Buku13'. It includes attributes for 'judul', 'pengarang', 'halaman', 'stok', and 'harga'. There are four methods: 'tampilInformasi()' for displaying book details, 'terjual()' for handling book sales with stock checks, 'restock()' for increasing stock, and 'gantiHarga()' for updating the price. The code is numbered from 1 to 35.

```
1  * Buku13
2  */
3  public class Buku13 {
4
5
6
7
8      String judul, pengarang;
9      int halaman, stok, harga;
10
11     void tampilInformasi() {
12         System.out.println("Judul: " + judul);
13         System.out.println("Pengarang: " + pengarang);
14         System.out.println("Jumlah halaman: " + halaman);
15         System.out.println("Sisa stok: " + stok);
16         System.out.println("Harga: Rp " + harga);
17     }
18
19     void terjual(int jml) {
20         if (stok > 0) {
21             stok -= jml;
22             System.out.println(jml + " buku terjual. Sisa stok: " + stok);
23         } else {
24             System.out.println("Stok habis. Tidak dapat melakukan penjualan.");
25         }
26     }
27
28     void restock(int jml) {
29         stok += jml;
30     }
31
32     void gantiHarga(int hrg) {
33         harga = hrg;
34     }
35 }
```

- **void restock(int jml):** Baris ini mendeklarasikan sebuah metode bernama restock yang menerima parameter berupa bilangan bulat jml (mewakili jumlah buku yang akan di-restock) dan tidak mengembalikan nilai apa pun (void).
- **stok += jml;:** Baris ini merupakan implementasi dari metode restock. Ini akan menambahkan nilai atribut stok (stok) dengan nilai parameter jml, efektif menambahkan jumlah buku yang ditentukan ke stok saat ini.
- **void gantiHarga(int hrg):** Baris ini mendeklarasikan sebuah metode bernama gantiHarga yang menerima parameter berupa bilangan bulat hrg (mewakili harga baru) dan tidak mengembalikan nilai apa pun (void).
- **harga = hrg;:** Baris ini merupakan implementasi dari metode gantiHarga. Ini akan mengganti nilai atribut harga dengan nilai dari parameter hrg, efektif memperbarui harga buku ke nilai baru yang ditentukan.

2.2.3 Pertanyaan

1. Pada class **BukuMain**, tunjukkan baris kode program yang digunakan untuk proses instansiasi!

Apa nama object yang dihasilkan? = `Buku13 bk1 = new Buku13();` Dan Objek yang dihasilkan adalah `bk1`

2. Bagaimana cara mengakses atribut dan method dari suatu objek? = menggunakan operator titik (.). Contohnya, pada baris kode program = `bk1.tampilinformasi();`

Kode di atas mengakses method `tampilinformasi()` dari objek `bk1`.

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

= Berbeda, karena pada pemanggilan pertama, informasi buku ditampilkan sebelum ada penjualan (`terjual(5)`) dan perubahan harga (`gantiHarga(60000)`), sedangkan pada pemanggilan kedua, informasi buku ditampilkan setelah ada penjualan dan perubahan harga. Jadi, hasil output mencerminkan kondisi terbaru dari objek `bk1`.

Hasil dari compile dan run program saya =

```
Judul: Today Ends Tomorrow Comes
Pengarang: Denanda Pratiwi
Jumlah halaman: 198
Sisa stok: 13
Harga: Rp 71000
5 buku terjual. Sisa stok: 8
Judul: Today Ends Tomorrow Comes
Pengarang: Denanda Pratiwi
Jumlah halaman: 198
Sisa stok: 8
Harga: Rp 60000
```

2.3.3 Pertanyaan

1. Pada class **Buku** di Percobaan 3, tunjukkan baris kode program yang digunakan untuk mendeklarasikan konstruktor berparameter! =

```
public Buku13(String jud, String pg, int hal, int stok, int har) {
    judul = jud;
    pengarang = pg;
    halaman = hal;
    this.stok = stok;
    harga = har;
}
```

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

```
Buku bk2 = new Buku(jud:"Self Reward", pg:"Maheera Ayesha", hal:160, stok:29, har:59000);
```

= Pada baris ini, objek bk2 dibuat menggunakan konstruktor berparameter dari class Buku13. Ini menunjukkan cara menggunakan konstruktor berparameter untuk membuat dan menginisialisasi objek Buku13 sekaligus.

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

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
    The constructor Buku13(String, String, int, int, int) is undefined

    at BukuMain13.main(BukuMain13.java:18)
PS C:\Users\ivanr\OneDrive\Desktop\Jobsheet2>
```

Hasilnya akan menjadi kesalahan kompilasi karena tidak ada konstruktor default yang dapat digunakan untuk membuat objek Buku13. Program perlu menyediakan nilai untuk setiap parameter konstruktor

4. Setelah melakukan instansiasi object, apakah method di dalam class **Buku** harus diakses secara berurutan? Jelaskan alasannya! = Tidak, karena Masing-masing method berfungsi secara independen dan dapat dipanggil secara terpisah tanpa perlu mengikuti urutan tertentu.

5. Buat object baru dengan nama **buku<NamaMahasiswa>** menggunakan konstruktor berparameter dari class **Buku!** =

```
public class BukuMahasiswa {
    Run | Debug
    public static void main(String[] args) {

        Buku13 bukuDenanda = new Buku13(jud:"Today Ends Tomorrow Come", pg:"Denanda Pratiwi", hal:198, stok:13, har:71000);
        bukuDenanda.tampilinformasi();

        Buku13 bukuMaheera = new Buku13(jud:"Self Reward", pg:"Maheera Ayesha", hal:160, stok:29, har:59000);
        bukuMaheera.tampilinformasi();
    }
}
```

2.4 Latihan Praktikum

Waktu : 150 Menit

= Pada latihan praktikum ini, saya menambahkan sesuai dengan yang diperintahkan di jobsheet ini,

Yang pertama saya tambahkan adalah harga total, Metode ini menggunakan operasi perkalian (*) untuk mengalikan nilai harga dengan stok guna menghitung total harga.

```
int hitungHargaTotal() {  
    return harga * stok;  
}
```

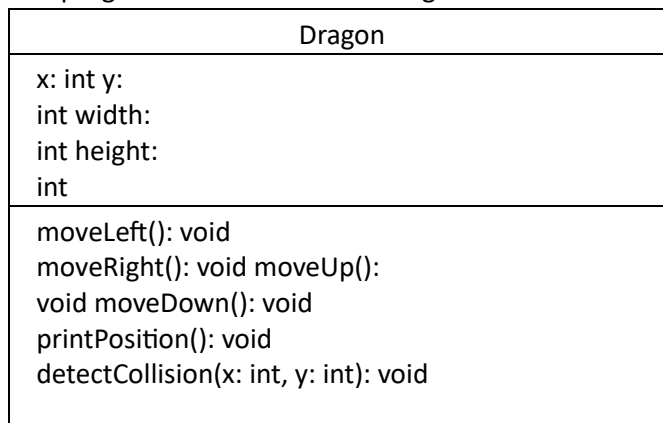
Yang kedua adalah menambahkan harga diskon yang sesuai dengan aturan jobsheet, Metode ini menggunakan beberapa operasi perbandingan (>, >=, <=) dan operasi perkalian (*) untuk menghitung diskon berdasarkan kondisi tertentu.

```
int hitungDiskon() {  
    int hargaTotal = hitungHargaTotal();  
  
    if (hargaTotal > 150000) {  
        return (int) (0.12 * hargaTotal);  
    } else if (hargaTotal >= 75000 && hargaTotal <= 150000) {  
        return (int) (0.05 * hargaTotal);  
    } else {  
        return 0;  
    }  
}
```

Dan yang terakhir adalah menghitung harga bayar nya , saya menggunakan operasi pengurangan (-) untuk menghitung harga bayar setelah diskon, dengan memanggil hitungHargaTotal() dan hitungDiskon().

```
int hitungHargaBayar() {  
    return hitungHargaTotal() - hitungDiskon();  
}
```

1. Buat program berdasarkan class diagram berikut ini!



Pada praktikum ini saya membuat program yang berjudul Dragon ini. Saya juga menggunakan file Main untuk memanggil nya. Dan yang akan saya jelaskan pertama adalah class dragon.

Pada class Dragon ini saya menggunakan atribut sebagai berikut :

- private int x: Menyimpan posisi koordinat x dari dragon.
- private int y: Menyimpan posisi koordinat y dari dragon.
- private int width: Menyimpan lebar (width) dari ruang gerak dragon.
- private int height: Menyimpan tinggi (height) dari ruang gerak dragon.

- Konstruktor Dragon13(int x, int y, int width, int height):

Menerima nilai awal untuk posisi dan ukuran dragon.

- Inisialisasi atribut x, y, width, dan height dengan nilai yang diberikan.
- Metode moveLeft(), moveRight(), moveUp(), moveDown():

Metode-metode ini menggerakkan dragon ke kiri, kanan, atas, dan bawah, masing-masing.

Sebelum menggerakkan, mereka melakukan pengecekan untuk memastikan dragon tetap berada dalam batas ruang gerak yang ditentukan.

Jika dragon akan keluar dari batas ruang gerak, metode detectCollision(int x, int y) dipanggil, yang saat ini hanya mencetak "Game Over".

- Metode detectCollision(int x, int y):

Metode ini memberikan pesan "Game Over" saat dragon berusaha keluar dari batas ruang gerak yang ditentukan.

- Metode printPosition():

Metode ini mencetak posisi saat ini dari dragon.

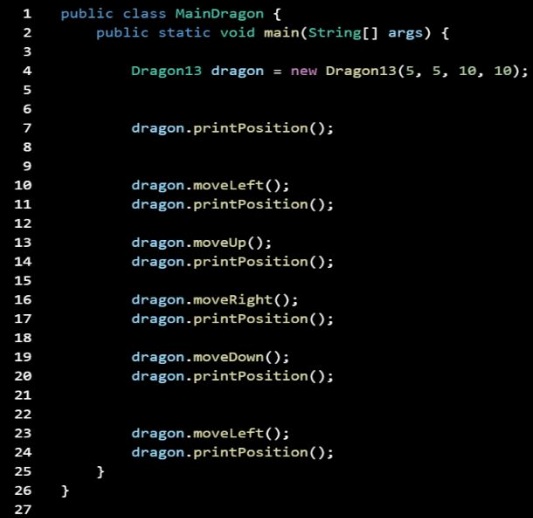
Dan untuk codingan saya adalah sebagai berikut=

```
1 public class Dragon13 {
2     private int x;
3     private int y;
4     private int width;
5     private int height;
6
7     public Dragon13(int x, int y, int width, int height) {
8         this.x = x;
9         this.y = y;
10        this.width = width;
11        this.height = height;
12    }
13
14    public void moveLeft() {
15        if (x - 1 >= 0) {
16            x--;
17        } else {
18            detectCollision(x - 1, y);
19        }
20    }
21
22    public void moveRight() {
23        if (x + 1 < width) {
24            x++;
25        } else {
26            detectCollision(x + 1, y);
27        }
28    }
29
30    public void moveUp() {
31        if (y - 1 >= 0) {
32            y--;
33        } else {
34            detectCollision(x, y - 1);
35        }
36    }
37
38    public void moveDown() {
39        if (y + 1 < height) {
40            y++;
41        } else {
42            detectCollision(x, y + 1);
43        }
44    }
45
46    // Specify the return type for detectCollision
47    public void detectCollision(int x, int y) {
48        System.out.println("Game Over");
49    }
50
51    public void printPosition() {
52        System.out.println("Dragon position: (" + x + ", " + y + ")");
53    }
54 }
55
```

Dan tidak lupa juga saya menambahkan Main untuk memanggil nya , apa fungsi nya? Ya fungsi dari Main di sini adalah sebagai pembuat objek dragon dengan koordinat awal (5, 5) dan ruang gerak dengan lebar 10 dan tinggi 10.

Dan juga untuk memanggil serangkaian metode untuk menggerakkan dragon ke berbagai arah dan mencetak posisinya setiap kali bergerak.

Untuk codingan dari main nya
adalah sebagai berikut =



```
1  public class MainDragon {  
2      public static void main(String[] args) {  
3  
4          Dragon13 dragon = new Dragon13(5, 5, 10, 10);  
5  
6  
7          dragon.printPosition();  
8  
9  
10         dragon.moveLeft();  
11         dragon.printPosition();  
12  
13         dragon.moveUp();  
14         dragon.printPosition();  
15  
16         dragon.moveRight();  
17         dragon.printPosition();  
18  
19         dragon.moveDown();  
20         dragon.printPosition();  
21  
22  
23         dragon.moveLeft();  
24         dragon.printPosition();  
25     }  
26 }  
27
```