LAPORAN 9
Pemrograman Berorientasi Objek

“Polimorfisme dan Kelas Abstrak”

Dosen Pengampu : Paulina Heruningsih Prima Rosa

DIBUAT OLEH :
Nama : Yohanis Celvin D.P.U.Pati
NIM : 245314033

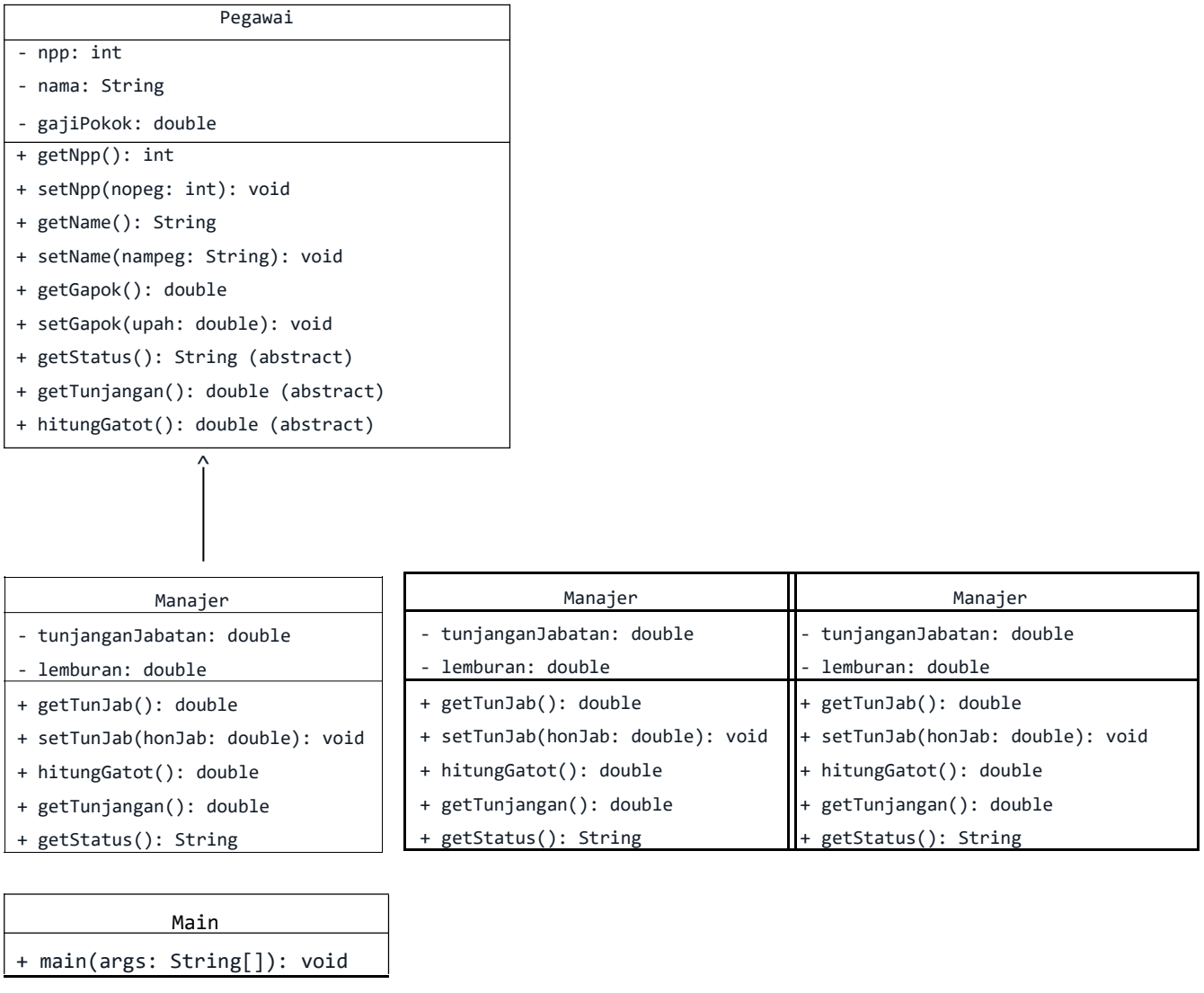
KELAS : BP

PROGRAM STUDI INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS SANATA DHARMA
YOGYAKARTA
2025

latihan kasus 1

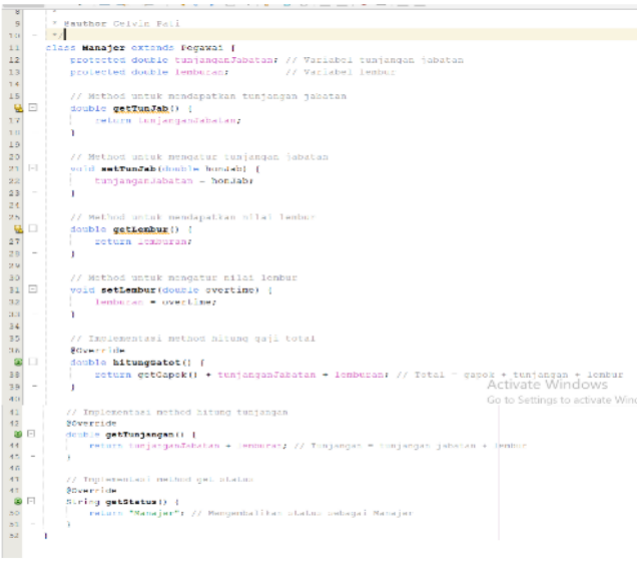
A. DIAGRAM CLASS

- Berikut Penggambaran diagram kelas:



C. LISTING PROGRAM

```
/**
 * @author Celvin Pati
 */
abstract class Pegawai {
    protected int npp;          // Nomor Pegawai
    protected String nama;      // Nama pegawai
    protected double gajiPokok; // Gaji pokok

    // Method untuk mendapatkan NPP
    int getNpp() {
        return npp;
    }

    // Method untuk mengatur NPP
    void setNpp(int nopeg) {
        npp = nopeg;
    }

    // Method untuk mendapatkan nama
    String getName() {
        return nama;
    }

    // Method untuk mengatur nama
    void setName(String nampeg) {
        nama = nampeg;
    }

    // Method untuk mendapatkan gaji pokok
    double getGapok() {
        return gajiPokok;
    }

    // Method untuk mengatur gaji pokok
    void setGapok(double gpok) {
        gajiPokok = gpok;
    }

    // Deklarasi method abstract yang harus diimplementasikan subclass
    abstract String getStatus();      // Untuk mendapatkan status jabatan
    abstract double gettunjangan();   // Untuk menghitung tunjangan
    abstract double hitungGatot();    // Untuk menghitung gaji total
}
```



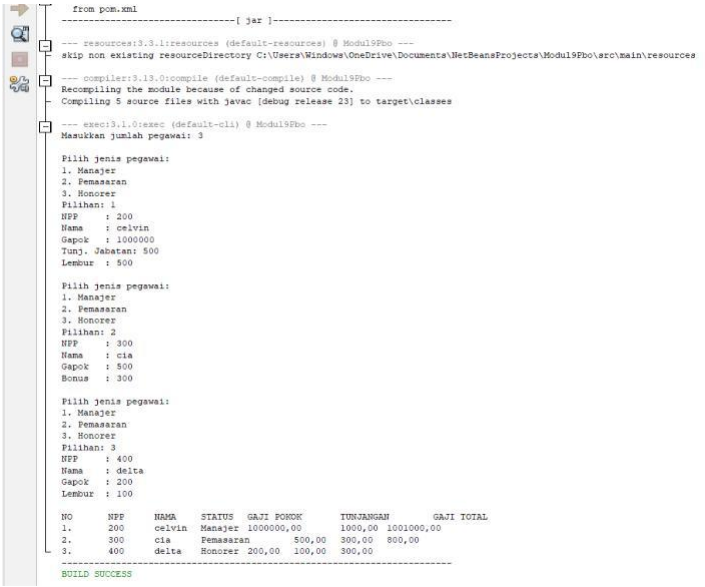
```
/**
 * @author Celvin Pati
 */
class Manajer extends Pegawai {
    protected double tunjanganJabatan; // Variabel tunjangan jabatan
    protected double lemburan;         // Variabel lembur

    // Method untuk mendapatkan tunjangan jabatan
    double getTunJab() {
        return tunjanganJabatan;
    }

    // Method untuk mengatur tunjangan jabatan
    void setTunJab(double honJab) {
        tunjanganJabatan = honJab;
    }

    // Method untuk mendapatkan nilai lembur
    double getLembur() {
        return lemburan;
    }

    // Method untuk mengatur nilai lembur
    void setLembur(double overtime) {
        lemburan = overtime;
    }

    // Implementasi method hitung gaji total
    @Override
    double hitungGatot() {
        return getGapok() + tunjanganJabatan + lemburan; // Total = gapok + tunjangan + lembur
    }

    // Implementasi method hitung tunjangan
    @Override
    double getTunjangan() {
        return tunjanganJabatan + lemburan; // Tunjangan = tunjangan jabatan + lembur
    }

    // Implementasi method get status
    @Override
    String getStatus() {
        return "Manajer"; // Mengembalikan status sebagai Manajer
    }
}
```

```
/**
 */
class Honorer extends Pegawai {
    protected double lemburan; // Variasi untuk menyimpan jumlah lembur

    // Method untuk mendapatkan nilai lembur
    double getLembur() {
        return lemburan;
    }

    // Method untuk mengatur nilai lembur
    void setLembur(double myLembur) {
        lemburan = myLembur;
    }

    // Implementasi method hitung gaji total
    @Override
    double hitungGatot() {
        return getGapok() + lemburan; // Gaji total = gaji pokok + lembur
    }

    // Implementasi method hitung tunjangan
    @Override
    double getTunjangan() {
        return lemburan; // Tunjangan dianggap sama dengan lembur
    }

    // Implementasi method get status
    @Override
    String getStatus() {
        return "Honorer"; // Mengembalikan status sebagai honorer
    }
}
```

```java
 * @author Calvin Pati
 */
class Pemasaran extends Pegawai {
    protected double bonus; // Variabel untuk menyimpan bonus

    // Method untuk mendapatkan bonus
    double getBonus() {
        return bonus;
    }

    // Method untuk mengatur bonus
    void setBonus(double myBonus) {
        bonus = myBonus;
    }

    // Implementasi method hitung gaji total
    @Override
    double hitungGatot() {
        return getGapok() + bonus; // Total = gapok + bonus
    }

    // Implementasi method hitung tunjangan
    @Override
    double getTunjangan() {
        return bonus; // Tunjangan dianggap sama dengan bonus
    }

    // Implementasi method get status
    @Override
    String getStatus() {
        return "Pemasaran"; // Mengembalikan status sebagai Pemasaran
    }
}
```

```java
 * @author Calvin Pati
 */
public class Main {
    public static void main(String[] args) {
        Pegawai[] karyawan = new Pegawai[10]; // Array untuk menyimpan objek pegawai
        int jumPeg = 0; // Variabel untuk jumlah pegawai

        Scanner input = new Scanner(System.in); // Objek Scanner untuk input

        // Input jumlah pegawai
        System.out.print("Masukkan jumlah pegawai: ");
        jumPeg = input.nextInt();
        input.nextLine(); // Membersihkan newline dari buffer

        // Loop untuk input data pegawai
        for(int i = 0; i < jumPeg; i++) {
            System.out.println("\nPilih jenis pegawai:");
            System.out.println("1. Manajer\n2. Pemasaran\n3. Honorer");
            System.out.print("Pilihan: ");
            int pilihan = input.nextInt();
            input.nextLine(); // Membersihkan buffer

            // Switch case untuk memilih jenis pegawai
            switch(pilihan) {
                case 1: // Case untuk Manajer
                    Manajer m = new Manajer();
                    System.out.print("NPP      : ");
                    m.setNpp(input.nextInt());
                    input.nextLine(); // Membersihkan buffer
                    System.out.print("Nama     : ");
                    m.setName(input.nextLine());
                    System.out.print("Gapok    : ");
                    m.setGapok(input.nextDouble());
                    System.out.print("Tunj. Jabatan: ");
                    m.setTunJab(input.nextDouble());
                    System.out.print("Lembur   : ");
                    m.setLembur(input.nextDouble());
                    karyawan[i] = m; // Simpan objek ke array
                    break;

                case 2: // Case untuk Pemasaran
                    Pemasaran p = new Pemasaran();
                    System.out.print("NPP      : ");
                    p.setNpp(input.nextInt());
                    input.nextLine();
                    System.out.print("Nama     : ");
                    p.setName(input.nextLine());
                    System.out.print("Gapok    : ");
                    p.setGapok(input.nextDouble());
                    System.out.print("Bonus    : ");
                    p.setBonus(input.nextDouble());
                    karyawan[i] = p;
                    break;

                case 3: // Case untuk Honorer
                    Honorer h = new Honorer();
                    System.out.print("NPP      : ");
                    h.setNpp(input.nextInt());
                    input.nextLine();
                    System.out.print("Nama     : ");
                    h.setName(input.nextLine());
                    System.out.print("Gapok    : ");
                    h.setGapok(input.nextDouble());
                    System.out.print("Lembur   : ");
                    h.setLembur(input.nextDouble());
                    karyawan[i] = h;
                    break;

                default: // Jika input tidak valid
                    System.out.println("Pilihan salah!");
                    i--; // Mengurangi counter untuk mengulang iterasi
            }
        }

        // Header Label output
        System.out.println("\nNO \tNPP \tNAMA \tSTATUS \tGAJI POKOK \tTUNJANGAN \tGAJI TOTAL");

        // Loop untuk menampilkan data pegawai
        for(int i = 0; i < jumPeg; i++) {
            Pegawai p = karyawan[i]; // Ambil objek pegawai dari array

            // Format output menggunakan printf untuk penyajian rapi
            System.out.printf("%d. \t%d \t%s \t%s \t%.2f \t%.2f \t%.2f%n",
                    (i+1),          // Nomor urut
                    p.getNpp(),     // NPP pegawai
                    p.getName(),    // Nama pegawai
                    p.getStatus(), // Status jabatan
                    p.getGapok(),   // Gaji pokok
                    p.getTunjangan(), // Tunjangan
                    p.hitungGatot() // Gaji total
            );
        }
    }
}
```

- output



D. KONSEP POLIMORFISME PROGRAM

1. Kelas abstrak dan metode abstrak:
   - Kelas “Pegawai” adalah kelas abstrak yang mendefinisikan metode abstrak getStarus(), getTunjangan(), dan hitungGatot(). Metode ini tidak memiliki implementasi di kelas “Pegawai”, tapi dideklarasikan untuk diimplementasikan oleh subclass.
   - Dengan mendeklarasikan metode sebagai abstrak, akan memaksa setiap subclass(‘Manajer’, ‘Pemasaran’, dan ‘Honorer’) untuk memberikan implementasi spesifik untuk metode tersebut.
2. Metode Override:
   - Setiap subclass(‘Manajer’, ‘Pemasaran’, dan ‘Honorer’) mengimplementasikan metode abstrak dari kelas ‘Pegawai’. Contoh:
     - Kelas ‘Manajer’ mengimplementasikan ‘getStatus()’ untuk mengembalikan ‘Manajer’.
     - Kelas ‘Pemasaran’ mengimplementasikan ‘getStatus()’ untuk mengembalikan ‘Pemasaran’.
     - Kelas ‘Honorer’ mengimplementasikan ‘getStatus()’ untuk mengembalikan ‘Honorer’.
   - Ini menunjukan bahwa mesikpun objek dari kelas tersebut dapat diperlakukan sebagai objek ‘Pegawai’, mereka meiliki perilaku yang berbeda sesuai dengan implementasi spesifik mereka.

3. Penggunakan Polimorfisme dalam Array:

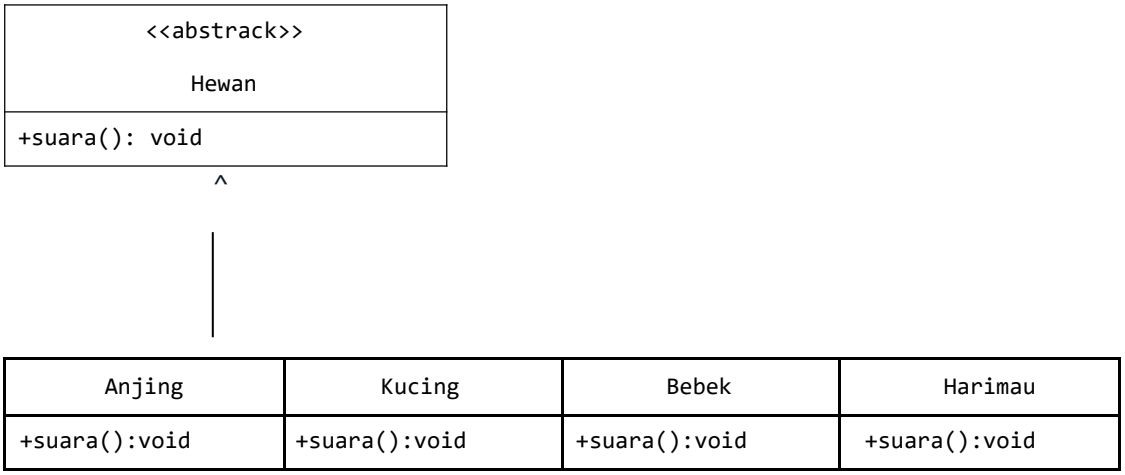
   - Dalam kelas ‘Main’, memiliki array ‘karyawa’ yang dideklarasikan sebagai ‘Pegawai[ ]’. Jadi bisah menyimpan objek dari berbagai subclass (’Manajer’, ‘Pemasaran’, dan ‘Honorer’) dalam array ini.

   - Ketika memanggil metode ‘getStatus()’, ‘getTunjangan()’, atau ‘hitungGatot()’ pada elemen array ‘karyawab’, maka java akan menetukan metode mana yang akan dipanggil sesuai dengan tipe objek yang sebenarnya.

Kasus 2

A. DIAGRAM CLASS

- Berikut penggambaran diagram kelas:



MainHewan
+ main(args: String[]): void

B. LISTING PROGRAM

```java
/**
 * @author Celvin Pati
 */
abstract class Hewan {
    // Deklarasi metode abstrak 'suara()' yang wajib diimplementasikan subclass
    abstract void suara();
}
// Kelas Anjing mewarisi (extends) dari kelas abstrak Hewan
class Anjing extends Hewan {
    @Override // Menandakan override metode dari parent class
    void suara() {
        System.out.println("gug gug"); // Implementasi suara anjing
    }
}
// Kelas Kucing mewarisi dari Hewan
class Kucing extends Hewan {
    @Override
    void suara() {
        System.out.println("meong-meong"); // Implementasi suara kucing
    }
}
// Kelas Bebek mewarisi dari Hewan
class Bebek extends Hewan {
    @Override
    void suara() {
        System.out.println("kwek-kwek"); // Implementasi suara bebek
    }
}
// Kelas baru Harimau mewarisi dari Hewan (tambahan sesuai permintaan)
class Harimau extends Hewan {
    @Override
    void suara() {
        System.out.println("raaar"); // Implementasi suara harimau
    }}
```

```java
 *
 * @author Celvin Pati
 */
class MainHewan {
    public static void main(String[] args) {
        Hewan kewan; // Deklarasi variabel bertipe abstrak Hewan

        // Polimorfisme: objek Anjing diakses melalui referensi Hewan
        kewan = new Anjing();
        kewan.suara(); // Memanggil metode suara() dari Anjing

        // Polimorfisme: objek Kucing diakses melalui referensi Hewan
        kewan = new Kucing();
        kewan.suara(); // Memanggil metode suara() dari Kucing

        // Polimorfisme: objek Bebek diakses melalui referensi Hewan
        kewan = new Bebek();
        kewan.suara(); // Memanggil metode suara() dari Bebek

        // Polimorfisme: objek Harimau diakses melalui referensi Hewan
        kewan = new Harimau();
        kewan.suara(); // Memanggil metode suara() dari Harimau
    }
}
```

- Output Program:

```
cd C:\Users\Windows\OneDrive\Documents\NetBeansProjects\Modul9Pbo; "JAVA_HOME=C:\\Program Files\\Java\\jdk-23" cmd /c "\"C:
Scanning for projects...

---------------------< com.mycompany:Modul9Pbo >----------------------
Building Modul9Pbo 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ Modul9Pbo ---
skip non existing resourceDirectory C:\Users\Windows\OneDrive\Documents\NetBeansProjects\Modul9Pbo\src\main\resources

--- compiler:3.13.0:compile (default-compile) @ Modul9Pbo ---
Recompiling the module because of changed source code.
Compiling 7 source files with javac [debug release 23] to target\classes

--- exec:3.1.0:exec (default-cli) @ Modul9Pbo ---
gug gug
meong-meong
kwek-kwek
raaar
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  3.484 s
Finished at: 2025-05-05T22:12:15+07:00
------------------------------------------------------------------------
```

C. KONSEP POLIMORFISME PROGRAM

1. Konsep polimorfisme dalam program Hewan

   - Polimorfisme adalah kemampuan objek untuk memiliki banyak bentuk (bentuk dinamis).
   - Pada program ini, variabel bertipe Hewan digunakan untuk merujuk ke objek dari berbagai subclass seperti Anjing, Kucing, Bebek, dan Harimau.
   - Method suara() didefinisikan secara abstrak di kelas Hewan dan diimplementasikan secara berbeda oleh masing-masing subclas
   - Saat method suara() dipanggil dari variabel Hewan, yang dieksekusi adalah versi milik subclass yang ditunjuk, bukan versi default dari superclass (karena memang abstrak).
   - Ini menunjukkan konsep runtime polymorphism**,** karena method yang dipanggil ditentukan saat program dijalankan, bukan saat dikompilasi.
   - Tujuan polimorfisme di sini adalah untuk:
     - Menyederhanakan kode.
     - Mempermudah ekspansi (misalnya menambah hewan baru tanpa ubah logika main()).
     - Mendukung prinsip Open/Closed (kode terbuka untuk ditambah, tertutup untuk diubah).

D. ANALISA PROGRAM

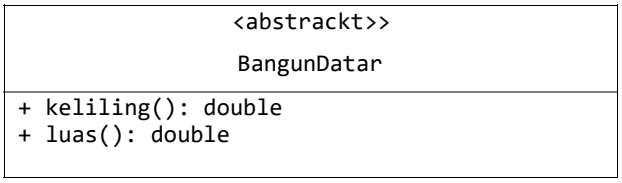
No	Syntax	penjelasan
1	abstract class Hewan { abstract void suara(); }	// Mendeklarasikan kelas abstrak Hewan dengan method abstrak suara() yang harus diimplementasikan oleh subclass.
2	class Anjing extends Hewan { void suara() { System.out.println("gug gug"); } }	//Kelas Anjing mewarisi Hewan dan mengimplementasikan method suara() untuk mencetak suara anjing.
3	class Kucing extends Hewan { void suara() { System.out.println("meon g-meong"); }	// Kelas Kucing mewarisi Hewan dan memberikan implementasi suara khas kucing.

4	class Bebek extends Hewan { void suara() { System.out.println("kwek-kwek"); } }	// Kelas Bebek meng-override method suara() untuk mencetak suara bebek.
5	class Harimau extends Hewan { void suara() { System.out.println("raaar"); }	//Kelas Harimau juga mengimplementasikan method suara() dengan suara khas harimau
6	Hewan h	//Mendeklarasikan variabel bertipe Hewan yang akan digunakan sebagai referensi objek turunan Hewan
7	h = new Anjing(); h.suara()	//Polimorfisme: objek Anjing diakses melalui referensi Hewan, lalu method suara() dipanggil.
8	h = new Kucing(); h.suara();	//Sama seperti sebelumnya, tapi objeknya Kucing.
9	h = new Bebek(); h.suara()	//Sama, tetapi untuk objek Bebek.
10	h = new Harimau(); h.suara();	//Sama, namun untuk objek Harimau.

Kasus 3

A. DIAGRAM CLASS

- Gambaran diagram class:



SegiEmpat	Segitiga	Lingkaran	MainB
- panjang: double - lebar: double	- sisiA: double - sisiB: double - sisiC: double	-jariJari: double	
+ SegiEmpat(panjang: double, lebar: double) + keliling(): double + luas(): double + toString(): String	+ Segitiga(a: double, b: double, c: double) + keliling(): double + luas(): double + toString(): String	+ Lingkaran(jariJari: double) + keliling(): double + luas(): double + toString(): String	+main(args: String[]): void

B. LISTING PROGRAM

```java
*/
abstract class BangunDatar {
    // Metode abstrak untuk menghitung keliling (wajib diimplementasikan subclass)
    public abstract double keliling();
    // Metode abstrak untuk menghitung luas (wajib diimplementasikan subclass)
    public abstract double luas();
}
```

```java
 *
 * @author Celvin Pati
 */
class SegiEmpat extends BangunDatar {
    // Atribut khusus SegiEmpat
    private double panjang;
    private double lebar;

    // Konstruktor untuk inisialisasi nilai panjang dan lebar
    public SegiEmpat(double panjang, double lebar) {
        this.panjang = panjang;
        this.lebar = lebar;
    }

    // Implementasi metode keliling untuk persegi panjang
    @Override
    public double keliling() {
        return 2 * (panjang + lebar); // Rumus keliling persegi panjang
    }

    // Implementasi metode luas untuk persegi panjang
    @Override
    public double luas() {
        return panjang * lebar; // Rumus luas persegi panjang
    }

    // Override toString untuk menampilkan nama bangun
    @Override
    public String toString() {
        return "Segi Empat";
    }
}
```

```java
 * @author Celvin Pati
 */
class Segitiga extends BangunDatar {
    // Atribut khusus Segitiga
    private double sisiA;
    private double sisiB;
    private double sisiC;

    // Konstruktor untuk inisialisasi ketiga sisi
    public Segitiga(double a, double b, double c) {
        this.sisiA = a;
        this.sisiB = b;
        this.sisiC = c;
    }

    // Implementasi metode keliling untuk segitiga
    @Override
    public double keliling() {
        return sisiA + sisiB + sisiC; // Jumlah semua sisi
    }

    // Implementasi metode luas menggunakan rumus Heron
    @Override
    public double luas() {
        double s = keliling() / 2; // Menghitung semi-perimeter
        return Math.sqrt(s * (s - sisiA) * (s - sisiB) * (s - sisiC)); // Rumus Heron
    }

    // Override toString untuk menampilkan nama bangun
    @Override
    public String toString() {
        return "Segitiga";
    }
```

```java
 *
 * @author Celvin Pati
 */
class Lingkaran extends BangunDatar {
    // Atribut khusus Lingkaran
    private double jariJari;

    // Konstruktor untuk inisialisasi jari-jari
    public Lingkaran(double jariJari) {
        this.jariJari = jariJari;
    }

    // Implementasi metode keliling lingkaran
    @Override
    public double keliling() {
        return 2 * Math.PI * jariJari; // 2πr
    }

    // Implementasi metode luas lingkaran
    @Override
    public double luas() {
        return Math.PI * jariJari * jariJari; // πr²
    }

    // Override toString untuk menampilkan nama bangun
    @Override
    public String toString() {
        return "Lingkaran";
    }
}
```

```java
 * @author Celvin Pati
 */
public class MainB {
    public static void main(String[] args) {
        // Membuat objek Scanner untuk input
        Scanner scanner = new Scanner(System.in);
        // Array untuk menyimpan 3 bangun datar
        BangunDatar[] bangunDatar = new BangunDatar[3];

        // Loop untuk input 3 bangun datar
        for (int i = 0; i < 3; i++) {
            System.out.println("\nBangun datar ke-" + (i+1));
            System.out.println("1. Segi Empat");
            System.out.println("2. Segitiga");
            System.out.println("3. Lingkaran");
            System.out.print("Pilih jenis bangun (1-3): ");

            // Membaca pilihan pengguna
            int pilihan = scanner.nextInt();

            // Switch case untuk memproses pilihan
            switch (pilihan) {
                case 1:  // Jika memilih segi empat
                    System.out.print("Masukkan panjang: ");
                    double p = scanner.nextDouble();
                    System.out.print("Masukkan lebar: ");
                    double l = scanner.nextDouble();
                    bangunDatar[i] = new SegiEmpat(p, l); // Membuat objek SegiEmpat
                    break;

                case 2:  // Jika memilih segitiga
                    System.out.print("Masukkan sisi 1: ");
                    double a = scanner.nextDouble();
                    System.out.print("Masukkan sisi 2: ");
                    double b = scanner.nextDouble();
                    System.out.print("Masukkan sisi 3: ");
                    double c = scanner.nextDouble();
                    bangunDatar[i] = new Segitiga(a, b, c); // Membuat objek Segitiga
                    break;

                case 3:  // Jika memilih lingkaran
                    System.out.print("Masukkan jari-jari: ");
                    double r = scanner.nextDouble();
                    bangunDatar[i] = new Lingkaran(r); // Membuat objek Lingkaran
                    break;

                default:  // Jika input tidak valid
                    System.out.println("Input tidak valid!");
                    i--; // Mengulang iterasi yang sama
                    break;
            }
        }

        // Menampilkan header output
        System.out.println("\n-------------------------------------------------");
        // Format header tabel
        System.out.printf("%-5s %-15s %-15s %-15s\n", "No.", "Bangun datar", "Keliling", "Luas");
        System.out.println("-------------------------------------------------");

        // Loop untuk menampilkan hasil perhitungan
        for (int i = 0; i < 3; i++) {
            BangunDatar b = bangunDatar[i]; // Mengakses elemen array
            // Mencetak data dengan format 2 angka desimal
            System.out.printf("%-5d %-15s %-15.2f %-15.2f\n",
                    (i+1), b.toString(), b.keliling(), b.luas());
        }

        // Garis penutup tabel
        System.out.println("-------------------------------------------------");
        scanner.close(); // Menutup scanner
    }
}
```


- output

```
--- compiler:3.13.0:compile (default-compile) @ Modul9Pbo ---
Recompiling the module because of changed source code.
Compiling 12 source files with javac [debug release 23] to target\classes

--- exec:3.1.0:exec (default-cli) @ Modul9Pbo ---

Bangun datar ke-1
1. Segi Empat
2. Segitiga
3. Lingkaran
Pilih jenis bangun (1-3): 1
Masukkan panjang: 3
Masukkan lebar: 4

Bangun datar ke-2
1. Segi Empat
2. Segitiga
3. Lingkaran
Pilih jenis bangun (1-3): 2
Masukkan sisi 1: 2
Masukkan sisi 2: 4
Masukkan sisi 3: 5

Bangun datar ke-3
1. Segi Empat
2. Segitiga
3. Lingkaran
Pilih jenis bangun (1-3): 3
Masukkan jari-jari: 4

=================================================
No.   Bangun datar    Keliling        Luas
-------------------------------------------------------------------------------
1     Segi Empat      14,00           12,00
2     Segitiga        11,00           3,80
3     Lingkaran       25,13           50,27
-------------------------------------------------------------------------------
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  53.368 s
Finished at: 2025-05-05T23:17:00+07:00
------------------------------------------------------------------------
```

C. PENJELASAN KONSEP POLIMORFISME

1. Konsep polimorfisme dalam program BangunDatar

- Polimorfisme (Polymorphism) berarti satu referensi objek bisa merujuk ke berbagai bentuk objek yang berbeda, dan menjalankan method yang sesuai dengan objek aktualnya.
- Di program ini, BangunDatar adalah kelas abstrak dengan method abstrak keliling() dan luas().
- Subclass seperti SegiEmpat, Segitiga, dan Lingkaran mengimplementasikan method keliling() dan luas() sesuai dengan rumus masing-masing bentuk.
- alam MainB, array BangunDatar[] bangunDatar = new BangunDatar[3]; digunakan untuk menyimpan berbagai objek (SegiEmpat, Segitiga, Lingkaran) dalam satu tipe umum: BangunDatar.
- Method keliling() dan luas() yang dipanggil adalah milik objek yang sebenarnya (misal Segitiga), bukan milik BangunDatar. Ini disebut dynamic binding.
- Ini menunjukkan polimorfisme runtime, yaitu method yang dipilih saat program dijalankan, tergantung pada tipe objek sebenarnya, bukan tipe referensinya.
- Manfaat polimorfisme dalam program ini:
    - Kode lebih fleksibel: satu array bisa menampung semua jenis bangun datar.
    - Mudah diperluas: jika nanti ditambah Trapesium, cukup tambahkan subclass baru tanpa ubah main().

D. ANALISIS

No	Syntax	penjelasan
1	abstract class BangunDatar { public abstract double keliling(); public abstract double luas(); }	// Mendeklarasikan kelas abstrak BangunDatar dengan dua method abstrak keliling() dan luas() yang wajib diimplementasikan oleh subclass.
2	class SegiEmpat extends BangunDatar { ... }	//Kelas SegiEmpat mewarisi BangunDatar dan mengimplementasikan method keliling() dan luas() sesuai rumus persegi panjang.
3	class Segitiga extends BangunDatar { ... }	//Kelas Segitiga juga merupakan turunan BangunDatar dan mengimplementasikan keliling() serta luas() menggunakan rumus Heron.
4	class Lingkaran extends BangunDatar { ... }	// Kelas Lingkaran mewarisi BangunDatar dan mengimplementasikan keliling() dan luas() berdasarkan rumus lingkaran.
5	BangunDatar[] bangunDatar = new BangunDatar[3];	//Mendeklarasikan array bertipe BangunDatar untuk menyimpan berbagai objek SegiEmpat, Segitiga, dan Lingkaran menggunakan prinsip polimorfisme
6	bangunDatar[i] = new SegiEmpat(p, l);	//Polimorfisme: Objek SegiEmpat diassign ke array bertipe BangunDatar.
7	bangunDatar[i] = new Segitiga(a, b, c);	//Polimorfisme: Objek Segitiga diassign ke array bertipe BangunDatar.
8	bangunDatar[i] = new Lingkaran(r);	//Polimorfisme: Objek Lingkaran diassign ke array bertipe BangunDatar.
9	keliling() dan luas() dalam	//Memanggil method keliling()

	loop for	dan luas() berdasarkan objek nyata (SegiEmpat, Segitiga, Lingkaran) melalui referensi BangunDatar (polimorfisme).

10	System.out.printf(...)	//Menampilkan informasi setiap objek bangun datar, termasuk nama, keliling, dan luas, dengan memanfaatkan toString() dari masing-masing kelas.