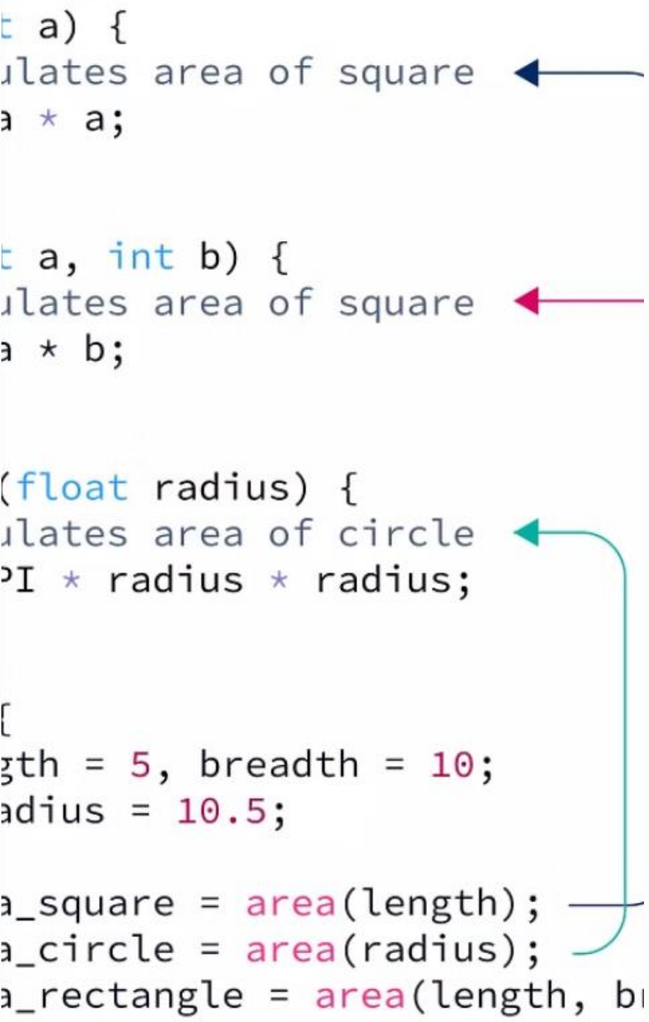
Polimorfisme : Fleksibilitas dalam Pemrograman Berorientasi Objek

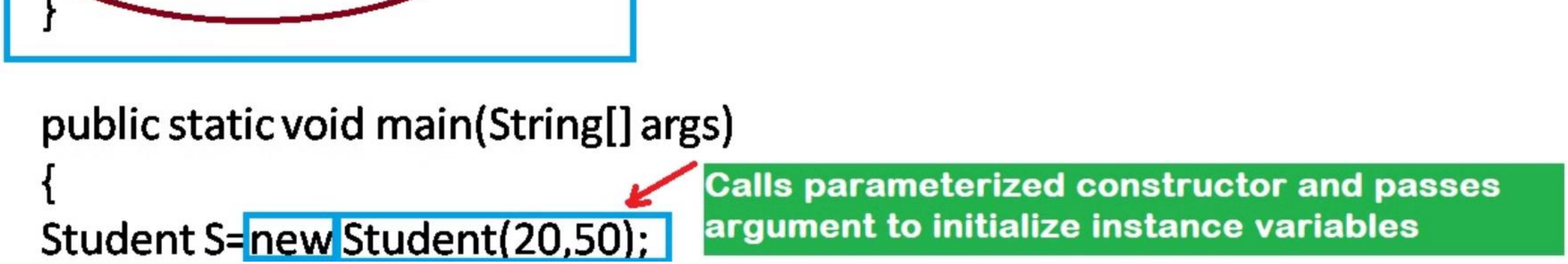
Polimorfisme adalah konsep penting dalam pemrograman berorientasi objek (OOP) yang memungkinkan objek dari kelas yang berbeda untuk diperlakukan sebagai objek dari kelas dasar yang sama. Ini memberikan fleksibilitas dan kekuatan dalam pengembangan aplikasi, memungkinkan penggunaan metode yang sama pada objek yang berbeda dengan perilaku yang sesuai.



SCALER Topics

Kelompok 4

Muhammad Abshar Hakim	20230810120
Rio Andika Andriansyah	20230810155
Yohanes Aditya Krismawan	20230810070



Konstruktor : Inisialisasi Objek dengan Mudah

Definisi Konstruktor

Konstruktor adalah metode khusus dalam kelas yang digunakan untuk menginisialisasi objek. Konstruktor memiliki nama yang sama dengan nama kelas dan tidak memiliki tipe pengembalian (return type).

Peran Konstruktor

Konstruktor dapat memiliki parameter untuk mengatur nilai awal dari atribut objek. Ini memungkinkan objek untuk langsung digunakan dengan nilai yang tepat tanpa perlu pengaturan tambahan.

Manfaat Konstruktor

Konstruktor membantu meningkatkan konsistensi dan kemudahan penggunaan objek, serta mengurangi kemungkinan kesalahan dalam inisialisasi objek.

Override : Menyesuaikan Perilaku Objek Objek

Definisi Override

Override adalah konsep dalam OOP yang memungkinkan subclass untuk menyediakan implementasi spesifik dari metode yang sudah didefinisikan di superclass.

Aturan Override

Metode yang dioverride harus memiliki nama yang sama, tipe pengembalian yang sama, dan parameter yang sama dengan metode di superclass.

Manfaat Override

Override memungkinkan subclass untuk menyesuaikan perilaku objek sesuai dengan kebutuhan mereka, meningkatkan fleksibilitas dan kemampuan adaptasi aplikasi.

Overloading : Meningkatkan Fleksibilitas Metode

1 Definisi Overloading

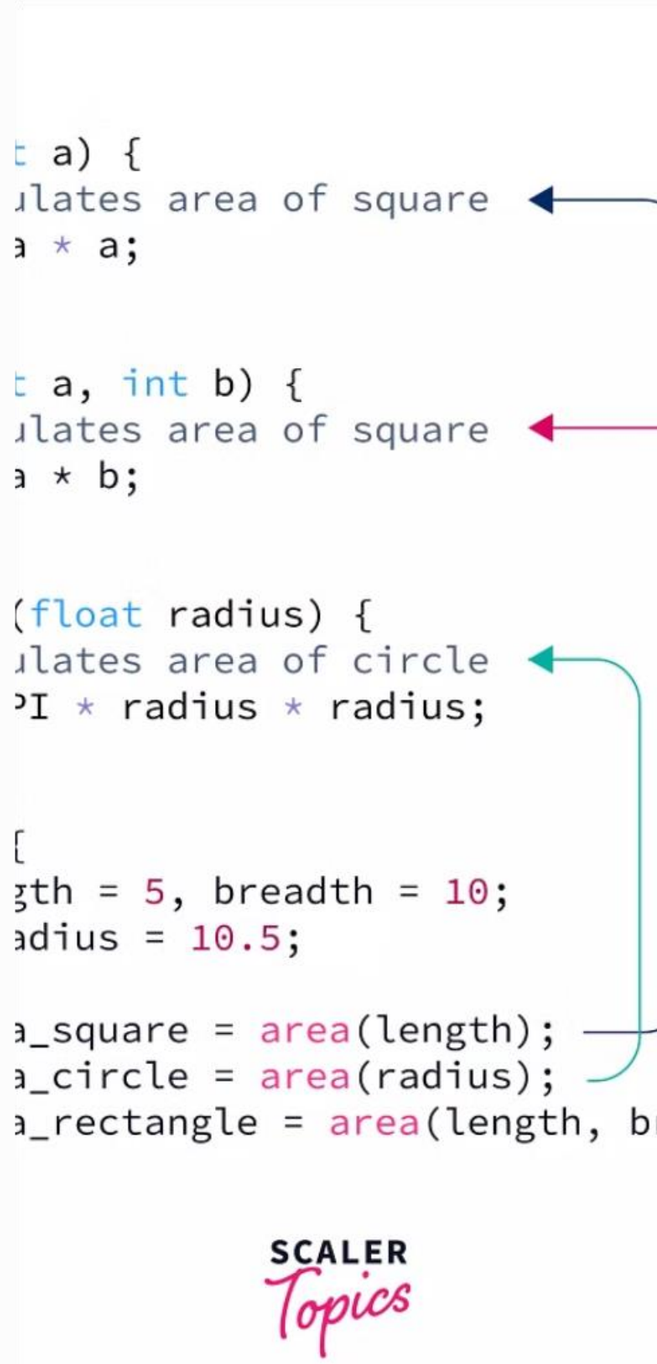
Overloading adalah konsep dalam OOP yang memungkinkan sebuah kelas memiliki lebih dari satu metode dengan nama yang sama tetapi dengan parameter yang berbeda.

2 Tujuan Overloading

Overloading membantu meningkatkan keterbacaan dan fleksibilitas kode dengan menyediakan berbagai versi metode yang dapat digunakan sesuai kebutuhan.

3 Contoh Overloading

Misalnya, sebuah kelas dapat memiliki metode info() dengan parameter berbeda untuk menampilkan informasi yang relevan.



SCALER Topics

Kode Program

Class Elektronik

```
import javax.swing.JOptionPane;

class Elektronik {
    protected String merek;
    protected int daya;
    protected int tahun;

    public Elektronik(String merek, int daya, int tahun) {
        this.merek = merek;
        this.daya = daya;
        this.tahun = tahun;
    }

    public void nyalakan() {
        JOptionPane.showMessageDialog(null, "Elektronik dinyalakan.");
    }

    public void info(String tambahan) {
        JOptionPane.showMessageDialog(null, "Merek: " + merek + "\nDaya: " + daya + " watt\nTahun: " + tahun + "\n" + tambahan);
    }
}
```

Penjelasan Code

Class Elektronik merupakan Superclass.

Atribut : merek, daya, dan tahun yang menggambarkan detail dasar dari perangkat elektronik.

Konstruktor : Menginisialisasi nilai atribut merek, daya, dan tahun.

Method nyalakan : Menampilkan pesan bahwa perangkat elektronik dinyalakan.

Metode info : Menampilkan informasi tentang perangkat elektronik dan tambahan informasi.

Kode Program

Class Kulkas

```
class Kulkas extends Elektronik {
    private int kapasitas;

    public Kulkas(String merek, int daya, int tahun, int kapasitas) {
        super(merek, daya, tahun);
        this.kapasitas = kapasitas;
    }

    @Override
    public void nyalakan() {
        JOptionPane.showMessageDialog(null, "Kulkas " + merek + " dengan kapasitas " + kapasitas + " liter dinyalakan.");
    }

    @Override
    public void info(String tambahan) {
        JOptionPane.showMessageDialog(null, "Merek: " + merek + "\nDaya: " + daya + " watt\nTahun: " + tahun + "\nKapasitas: " + kapasitas + " liter\n"
+ tambahan);
    }
}
```

Penjelasan Code

Class Kulkas merupakan turunan dari kelas Elektronik.

Atribut Tambahan : kapasitas untuk menggambarkan kapasitas kulkas dalam liter.

Konstruktor : Menginisialisasi nilai atribut dari kelas dasar dan atribut kapasitas.

Override Metode nyalakan : Menampilkan pesan bahwa kulkas dengan merek dan kapasitas tertentu dinyalakan.

Override Metode info : Menampilkan informasi tentang kulkas, termasuk kapasitasnya.

Kode Program

Class Televisi

```
class Televisi extends Elektronik {
    private int ukuranLayar;

    public Televisi(String merek, int daya, int tahun, int ukuranLayar) {
        super(merek, daya, tahun);
        this.ukuranLayar = ukuranLayar;
    }

    @Override
    public void nyalakan() {
        JOptionPane.showMessageDialog(null, "Televisi " + merek + " dengan ukuran layar " + ukuranLayar + " inci dinyalakan.");
    }

    @Override
    public void info(String tambahan) {
        JOptionPane.showMessageDialog(null, "Merek: " + merek + "\nDaya: " + daya + " watt\nTahun: " + tahun + "\nUkuran Layar: " + ukuranLayar +
" inci\n" + tambahan);
    }
}
```

Penjelasan Code

Class Televisi merupakan turunan dari kelas Elektronik.

Atribut Tambahan : kapasitas untuk menggambarkan kapasitas kulkas dalam liter.

Konstruktor : Menginisialisasi nilai atribut dari kelas dasar dan atribut kapasitas.

Override Metode nyalakan : Menampilkan pesan bahwa kulkas dengan merek dan kapasitas tertentu dinyalakan.

Override Metode info : Menampilkan informasi tentang kulkas, termasuk kapasitasnya.

Kode Program

Class Mesin Cuci

```
class MesinCuci extends Elektronik {
    private int kapasitas;

    public MesinCuci(String merek, int daya, int tahun, int kapasitas) {
        super(merek, daya, tahun);
        this.kapasitas = kapasitas;
    }

    @Override
    public void nyalakan() {
        JOptionPane.showMessageDialog(null, "Mesin Cuci " + merek + " dengan kapasitas " + kapasitas + " kg dinyalakan.");
    }

    @Override
    public void info(String tambahan) {
        JOptionPane.showMessageDialog(null, "Merek: " + merek + "\nDaya: " + daya + " watt\nTahun: " + tahun + "\nKapasitas: " + kapasitas + " kg\n" + tambahan);
    }
}
```

Penjelasan Code

Class Mesin Cuci merupakan turunan dari kelas Elektronik.

Atribut Tambahan : kapasitas untuk menggambarkan Kapasitas Mesin Cuci dalam Kg.

Konstruktor : Menginisialisasi nilai atribut dari kelas dasar dan atribut Kapasitas.

Override Metod nyalakan : Menampilkan pesan bahwa Mesin cuci dengan merek dan Kapasitas tertentu dinyalakan.

Override Method info : Menampilkan informasi tentang mesin cuci, termasuk Kapasitasnya.

Kode Program

Class Main

```
class Main {
    public static void main(String[] args) {
        Elektronik elektronik = null;

        String[] options = {"Kulkas", "Televisi", "Mesin Cuci"};
        int choice = JOptionPane.showOptionDialog(null, "Pilih jenis elektronik:", "Pilihan Elektronik",
                JOptionPane.DEFAULT_OPTION, JOptionPane.INFORMATION_MESSAGE, null, options, options[0]);
        if(choice == JOptionPane.CLOSED_OPTION){
            System.exit(0);
        }

        String merek = JOptionPane.showInputDialog("Masukkan merek:");
        int daya = Integer.parseInt(JOptionPane.showInputDialog("Masukkan daya (watt):"));
        int tahun = Integer.parseInt(JOptionPane.showInputDialog("Masukkan tahun:"));

        switch (choice) {
            case 0:
                int kapasitas = Integer.parseInt(JOptionPane.showInputDialog("Masukkan kapasitas (liter):"));
                elektronik = new Kulkas(merek, daya, tahun, kapasitas);
                break;
```

Kode Program

Lanjutan Class Main

```
            case 1:
                int ukuranLayar = Integer.parseInt(JOptionPane.showInputDialog("Masukkan ukuran layar (inci):"));
                elektronik = new Televisi(merek, daya, tahun, ukuranLayar);
                break;
            case 2:
                kapasitas = Integer.parseInt(JOptionPane.showInputDialog("Masukkan kapasitas (kg):"));
                elektronik = new MesinCuci(merek, daya, tahun, kapasitas);
                break;
            default:
                System.exit(0);

        }

        elektronik.nyalakan();
        elektronik.info("");
    }
}
```

Penjelasan Code

Metode main : Bagian yang menjalankan program.

Menampilkan dialog untuk memilih antara Kulkas, Televisi, atau Mesin Cuci meminta pengguna memasukkan detail seperti merek, daya, tahun, kapasitas (untuk kulkas), ukuran layar (untuk televisi) dan kapasitas(KG)(untuk Mesin Cuci.

Membuat objek dari kelas Kulkas, Televisi, atau Mesin Cuci berdasarkan input pengguna.

Memanggil metode nyalakan dan info untuk menampilkan pesan yang sesuai.

Aplikasi Konsep OOP : Kelas Elektronik

Kelas Elektronik

Kelas Elektronik memiliki atribut dasar seperti merek, daya, dan tahun pembuatan. Kelas ini juga memiliki metode untuk menyalakan perangkat dan menampilkan informasinya.

Polimorfisme

Objek dari kelas Kulkas, Televisi dan Mesin Cuci dapat diperlakukan sebagai objek dari kelas Elektronik, menunjukkan konsep polimorfisme. Ini memungkinkan penggunaan metode yang sama pada objek yang berbenis.

1

2

3

Kelas Turunan

Kelas Kulkas, Televisi dan Mesin Cuci merupakan kelas turunan dari Elektronik. Mereka menambahkan atribut spesifik seperti kapasitas dan ukuran layar, serta mengoverride metode untuk menampilkan informasi yang sesuai.

Interaksi Pengguna dengan GUI

Dialog Input

Program menggunakan JOptionPane untuk berinteraksi dengan pengguna, meminta input seperti merek, daya, dan ukuran perangkat.

Tampilan Informasi

Informasi tentang perangkat elektronik yang dipilih, seperti merek, daya, tahun, dan kapasitas/ukuran layar, ditampilkan dalam kotak pesan.

Antarmuka Pengguna

Penggunaan GUI membuat program lebih user-friendly dan interaktif, memudahkan pengguna dalam menggunakan aplikasi.

Manfaat Konsep OOP

Modularitas

Pemrograman berorientasi objek memungkinkan pengembangan aplikasi yang modular, dengan komponen-komponen yang dapat digunakan kembali.

Fleksibilitas

Konsep-konsep seperti inheritance, overriding, dan polimorfisme meningkatkan fleksibilitas dalam pengembangan aplikasi.

Kemudahan Pemeliharaan

Struktur kode yang jelas dan penggunaan kembali komponen memudahkan pemeliharaan dan pengembangan aplikasi dalam jangka panjang.

Efisiensi

Pemrograman berorientasi objek memungkinkan pengembangan aplikasi yang lebih efisien dan produktif.

Kesimpulan : Kekuatan OOP

1

Inheritance

Kelas turunan mewarisi atribut dan metode dari kelas dasar, memungkinkan penggunaan kembali kode yang ada.

2

Overriding

Subclass dapat menyediakan implementasi spesifik dari metode, menyesuaikan perilaku objek sesuai kebutuhan.

3

Polimorfisme

Objek dari kelas yang berbeda dapat diperlakukan sebagai objek dari kelas dasar yang sama, meningkatkan fleksibilitas.

Terima Kasih

Jika ada yang ingin ditanyakan, silahkan bertanya !