PEWARISAN SIFAT OBYEK

MUH. IZZUDDIN MAHALI, M.CS.

MEMBUAT KELAS TURUNAN (*SUBCLASS*)

- JAVA MENYEDIAKAN KATA KUNCI **`EXTENDS`** YANG DIGUNAKAN UNTUK PENURUNAN TERHADAP KELAS.
- DALAM TERMINOLOGI JAVA, KELAS INDUK YANG DITURUNKAN DISEBUT *SUPERCLASS*, SEDANGKAN KELAS BARU HASIL TURUNAN DISEBUT *SUBCLASS*.
- BENTUK UMUMNYA:

```
CLASS NAMA-SUBCLASS EXTENDS NAMA-SUPERCLASS{
//BADAN KELAS
}
```

MEMBUAT KELAS TURUNAN (*SUBCLASS*)

CONTOH

DEMOKELASTURUNAN.JAVA

```
CLASS A {
  PRIVATE INT A;

  PUBLIC VOID SETA(INT NILAI) {
    A = NILAI;
  }

  PUBLIC INT GETA() {
    RETURN A;
  }
}

CLASS B EXTENDS A {
  PRIVATE INT B;

  PUBLIC VOID SETB(INT NILAI) {
    B = NILAI;
  }

  PUBLIC INT GETB() {
    RETURN B;
  }
}
```

```
CLASS C EXTENDS B {
  PRIVATE INT C;

  PUBLIC VOID SETC(INT NILAI) {
    C = NILAI;
  }

  PUBLIC INT GETC() {
    RETURN C;
  }
}


CLASS DEMOKELASTURUNAN2 {
  PUBLIC STATIC VOID MAIN(STRING[] ARGS) {

C OBJ = NEW C();

OBJ.SETA(100);
    OBJ.SETB(200);
    OBJ.SETC(300);

    SYSTEM.OUT.PRINTLN("NILAI A : " + OBJ.GETA());
    SYSTEM.OUT.PRINTLN("NILAI B : " + OBJ.GETB());
    SYSTEM.OUT.PRINTLN("NILAI C : " + OBJ.GETC());
  }
}
```

TINGKAT AKSES **PROTECTED**

MELALUI TINGKAT AKSES **PROTECTED**, DATA DAPAT DIAKSES OLEH SEMUA KELAS YANG MEMILIKI HUBUNGAN TURUNAN, TAPI LINGKUNGAN LUAR TETAP TIDAK DIBERI HAK UNTUK MENGAKSES DATA TERSEBUT.

TINGKAT AKSES **PROTECTED**

CONTOH:

- DEMOPROTECTED1.JAVA
- DEMOPROTECTED2.JAVA

TINGKAT AKSES **PROTECTED**

DEMOPROTECTED1.JAVA

```
CLASS A {
  PRIVATE INT A;
  PROTECTED VOID SETA(INT NILAI) {
    A = NILAI;
  }

PROTECTED INT GETA() {
    RETURN A;
  }
}

CLASS B EXTENDS A {
  PRIVATE INT B;

B(INT NILAIA, INT NILAIB) {
    //A = NILAIA;  // SALAH, KARENA A TIDAK DIKENALI
DI SINI
    SETA(NILAIA);
    B = NILAIB;
  }

  PUBLIC VOID SHOWDATA() {
    SYSTEM.OUT.PRINTLN("NILAI A : " + GETA());
    SYSTEM.OUT.PRINTLN("NILAI B : " + B);
  }
}
```

```
CLASS C {
  PRIVATE INT C;

  PUBLIC VOID SETC(INT NILAI) {
    //SETA(10);
    C = NILAI;
  }

  PUBLIC INT GETC() {
    RETURN C;
  }

  PUBLIC VOID SHOWC() {
    //SYSTEM.OUT.PRINTLN("NILAI A : " + GETA());
              // SALAH
    SYSTEM.OUT.PRINTLN("NILAI C : " + C);
  }
}

CLASS DEMOPROTECTED1 {
  PUBLIC STATIC VOID MAIN(STRING[] ARGS) {

    // MELAKUKAN INSTANSIASI TERHADAP KELAS B
    B OBJ = NEW B(40, 50);

    OBJ.SHOWDATA();

    OBJ.SETA(100);
    SYSTEM.OUT.PRINTLN("NILAI A : " + OBJ.GETA());
  }
}
```

TINGKAT AKSES **PROTECTED**

DEMOPROTECTED2.JAVA

```
CLASS KOTAK {
  PROTECTED DOUBLE PANJANG;
  PROTECTED DOUBLE LEBAR;
  PROTECTED DOUBLE TINGGI;

  KOTAK() {
    PANJANG = LEBAR = TINGGI = 0;
  }

  KOTAK(INT P, INT L, INT T) {
    PANJANG = P;
    LEBAR = L;
    TINGGI = T;
  }

  PUBLIC DOUBLE HITUNGVOLUME() {
    RETURN (PANJANG * LEBAR * TINGGI);
  }
}

CLASS KOTAKPEJAL EXTENDS KOTAK {
  PRIVATE DOUBLE BERAT;

  KOTAKPEJAL(INT P, INT L, INT T, INT B) {
    PANJANG = P;
    LEBAR = L;
    TINGGI = T;
    BERAT = B;
  }

  PUBLIC DOUBLE GETBERAT() {
    RETURN BERAT;
  }
}
```

```
CLASS DEMOPROTECTED2 {
  PUBLIC STATIC VOID MAIN(STRING[] ARGS) {

    KOTAKPEJAL K = NEW KOTAKPEJAL(4, 3, 2, 1);

    SYSTEM.OUT.PRINTLN("VOLUME K\T : " + K.HITUNGVOLUME());
    SYSTEM.OUT.PRINTLN("BERAT K\T  : " + K.GETBERAT());
  }
}
```

MENGGUNAKAN KATA KUNCI **SUPER**

- CONSTRUCTOR YANG TERDAPAT PADA KELAS INDUK DAPAT DIPANGGIL DARI KELAS TURUNAN MENGGUNAKAN KATA KUNCI **SUPER**
- BENTUK UMUM PEMANGGILANNYA:

  SUPER(DAFTAR-PARAMETER)
- ***DAFTAR-PARAMETER*** = DAFTAR PARAMETER YANG DIDEFINISIKAN CONSTRUCTOR KELAS INDUK
- DALAM CONSTRUCTOR KELAS TURUNAN, **SUPER()** HARUS DITEMPATKAN PADA BAGIAN AWAL (BARIS PERTAMA).

MENGGUNAKAN `SUPER` UNTUK MEMANGGIL CONSTRUCTOR KELAS INDUK

CONTOH DEMOSUPER1.JAVA

```
CLASS KOTAK {
  PROTECTED DOUBLE PANJANG;
  PROTECTED DOUBLE LEBAR;
  PROTECTED DOUBLE TINGGI;

  KOTAK() {
    PANJANG = LEBAR = TINGGI = 0;
  }

  KOTAK(INT P, INT L, INT T) {
    PANJANG = P;
    LEBAR = L;
    TINGGI = T;
  }

  PUBLIC DOUBLE HITUNGVOLUME() {
    RETURN (PANJANG * LEBAR * TINGGI);
  }
}
```

```
CLASS KOTAKPEJAL EXTENDS KOTAK {
  PRIVATE DOUBLE BERAT;

  KOTAKPEJAL(INT P, INT L, INT T, INT B) {

    SUPER(P, L, T);  // MEMANGGIL CONSTRUCTOR KELAS KOTAK

    BERAT = B;
  }

  PUBLIC DOUBLE GETBERAT() {
    RETURN BERAT;
  }
}

CLASS DEMOSUPER1 {
  PUBLIC STATIC VOID MAIN(STRING[] ARGS) {

    // MELAKUKAN INSTANSIASI TERHADAP KELAS TURUNAN
    KOTAKPEJAL K = NEW KOTAKPEJAL(6, 5, 4, 2);

    SYSTEM.OUT.PRINTLN("VOLUME K : " + K.HITUNGVOLUME());
    SYSTEM.OUT.PRINTLN("BERAT K  : " + K.GETBERAT());
  }
}
```

MENGGUNAKAN `SUPER` UNTUK MENGAKSES ANGGOTA KELAS INDUK

CONTOH DEMOSUPER3.JAVA

```
CLASS A {
  PROTECTED INT A;
}

CLASS B EXTENDS A {
  PRIVATE INT A; // AKAN MENIMPA A YANG ADA DALAM KELAS A

  B(INT NILAI1, INT NILAI2) {
    SUPER.A = NILAI1; // A DI DALAM KELAS A
    A = NILAI2; // A DI DALAM KELAS B
  }

  PUBLIC VOID TAMPILKANNILAI() {
    SYSTEM.OUT.PRINTLN("NILAI A DI DALAM KELAS A : " + SUPER.A);
    SYSTEM.OUT.PRINTLN("NILAI A DI DALAM KELAS B : " + A);
  }
}

CLASS DEMOSUPER3 {
  PUBLIC STATIC VOID MAIN(STRING[] ARGS) {

    B OBJ = NEW B(121, 212);
    OBJ.TAMPILKANNILAI();
  }
}
```

MELAKUKAN *OVERRIDE* TERHADAP METHOD

- SAAT KITA MENDEFINISIKAN SUATU METHOD DI KELAS TURUNAN YANG NAMA DAN DAFTAR PARAMETERNYA SAMA PERSIS SEPERTI YANG TERDAPAT DI KELAS INDUK, MAKA DIKATAKAN KITA TELAH MELAKUKAN OVERRIDE (MENGESAMPINGKAN) METHOD YANG TERSIMPAN DI KELAS INDUK.
- APABILA KITA MEMANGGIL METHOD YANG TELAH DI-OVERRIDE MELALUI OBJEK KELAS TURUNAN, MAKA YANG KAKAN DIEKSEKUSI ADALAH METHOD YANG TERDAPAT PADA KELAS TURUNAN.

MELAKUKAN *OVERRIDE* TERHADAP METHOD

CONTOH:

- DEMOOVERRIDE1.JAVA
- DEMOOVERRIDE2.JAVA

MENGGUNAKAN **SUPER** UNTUK MEMANGGIL CONSTRUCTOR KELAS INDUK

CONTOH DEMOOVERRIDE1.JAVA

```
CLASS A {
  PRIVATE INT A;

  PUBLIC VOID SETA(INT NILAI) {
    A = NILAI;
  }

  PUBLIC INT GETA() {
    RETURN A;
  }

  PUBLIC VOID TAMPILKANNILAI() {
    SYSTEM.OUT.PRINTLN("NILAI A: " + GETA());
  }
}
```

```
CLASS B EXTENDS A {
  PRIVATE INT B;

  PUBLIC VOID SETB(INT NILAI) {
    B = NILAI;
  }

  PUBLIC INT GETB() {
    RETURN B;
  }

  // MELAKUKAN OVERRIDE TERHADAP METHOD TAMPILKANNILAI()
  // YANG TERDAPAT PADA KELAS A
  PUBLIC VOID TAMPILKANNILAI() {
    SYSTEM.OUT.PRINTLN("NILAI B: " + GETB());
  }
}

CLASS DEMOOVERRIDE1 {
  PUBLIC STATIC VOID MAIN(STRING[] ARGS) {

    B OBJ = NEW B();

    OBJ.SETA(100);
    OBJ.SETB(200);

    // AKAN MEMANGGIL METHOD YANG TERDAPAT PADA KELAS B
    OBJ.TAMPILKANNILAI();
  }
}
```

MENGGUNAKAN **SUPER** UNTUK MEMANGGIL CONSTRUCTOR KELAS INDUK

CONTOH DEMOOVERRIDE2.JAVA

```
CLASS A {
  PRIVATE INT A;

  PUBLIC VOID SETA(INT NILAI) {
    A = NILAI;
  }

  PUBLIC INT GETA() {
    RETURN A;
  }

  PUBLIC VOID TAMPILKANNILAI() {
    SYSTEM.OUT.PRINTLN("NILAI A: " + GETA());
  }
}
```

```
CLASS B EXTENDS A {
  PRIVATE INT B;

  PUBLIC VOID SETB(INT NILAI) {
    B = NILAI;
  }

  PUBLIC INT GETB() {
    RETURN B;
  }

  // MELAKUKAN OVERRIDE TERHADAP METHOD TAMPILKANNILAI()
  // YANG TERDAPAT PADA KELAS A
  PUBLIC VOID TAMPILKANNILAI() {
    SUPER.TAMPILKANNILAI();  // MEMANGGIL METHOD DALAM KELAS A
    SYSTEM.OUT.PRINTLN("NILAI B: " + GETB());
  }
}

CLASS DEMOOVERRIDE2 {
  PUBLIC STATIC VOID MAIN(STRING[] ARGS) {

    B OBJ = NEW B();

    OBJ.SETA(100);
    OBJ.SETB(200);

    // AKAN MEMANGGIL METHOD YANG TERDAPAT PADA KELAS B
    OBJ.TAMPILKANNILAI();
  }
}
```

PERBEDAAN *OVERRIDE* DENGAN *OVERLOAD*

- OVERRIDE : MENDEFINISIKAN SUATU METHOD DALAM KELAS TURUNAN YANG MEMILIKI NAMA DAN DAFTAR PARAMETER SAMA PERSIS DENGAN YANG TERDAPAT PADA KELAS INDUK.
- OVERLOAD : MENDEFINISIKAN SUATU METHOD DI DALAM KELAS TURUNAN YANG NAMANYA SAMA DENGAN METHOD YANG TERDAPAT PADA KELAS INDUKNYA, **TAPI DAFTAR PARAMETERNNYA BERBEDA**.
- CONTOH : DEMOOVERLOADTURUNAN.JAVA

PERBEDAAN *OVERRIDE* DENGAN *OVERLOAD*

CONTOH DEMOOVERLOADTURUNAN.JAVA

```
CLASS INDUK {
  // MENDEFINISIKAN METHOD TEST() TANPA PARAMETER
  PUBLIC VOID TEST() {
    SYSTEM.OUT.PRINTLN("METHOD DI DALAM KELAS INDUK");
  }
}

CLASS TURUNAN EXTENDS INDUK {
  // MELAKUKAN OVERLOAD TERHADAP METHOD TEST(), BUKAN OVERRIDE.
  PUBLIC VOID TEST(STRING S) {
    SYSTEM.OUT.PRINTLN("METHOD DI DALAM KELAS TURUNAN");
    SYSTEM.OUT.PRINTLN("S : \"" + S + "\"");
  }
}

CLASS DEMOOVERLOADTURUNAN {
  PUBLIC STATIC VOID MAIN(STRING[] ARGS) {

    TURUNAN OBJ = NEW TURUNAN();

    // MEMANGGIL METHOD TEST() YANG TERDAPAT PADA KELAS INDUK
    OBJ.TEST();
    SYSTEM.OUT.PRINTLN();

    // MEMANGGIL METHOD TEST() YANG TERDAPAT PADA KELAS TURUNAN
    OBJ.TEST("CONTOH OVERLOAD PADA PROSES PEWARISAN");
  }
}
```

POLIMORFISME

- POLIMORFISME MENGIZINKAN KELAS INDUK UNTUK MENDEFINISIKAN SEBUAH METHOD *GENERAL* (BERSIFAT UMUM) UNTUK SEMUA KELAS TURUNANNYA, DAN SELANJUTNYA KELAS-KELAS TURUNAN DAPAT MEMPERBARUI IMPLEMENTASI DARI METHOD TERSEBUT SECARA LEBIH SPESIFIK SESUAI DENGAN KARAKTERISTIK MASING-MASING.
- PROSES OVERRIDE DIBENTUK AGAR JAVA DAPAT MENDUKUNG KONSEP POLIMORFISME.

POLIMORFISME

CONTOH:

- DEMOPOLIMORFISME1.JAVA
- DEMOPOLIMORFIME2.JAVA

POLIMORFISME

CONTOH DEMOPOLIMORFISME1.JAVA

```
CLASS PENYANYI {
  PUBLIC VOID BERNYANYI() {
    SYSTEM.OUT.PRINTLN("KARAKTERISTIK NADANYA " +
    "BELUM DIDEFINISIKAN");
  }
}

// MENDEFINISIKAN KELAS-KELAS TURUNAN DARI KELAS PENYANYI
CLASS PENYANYIJAZZ EXTENDS PENYANYI {

  // MELAKUKAN OVERRIDE TERHADAP METHOD BERNYANYI()
  PUBLIC VOID BERNYANYI() {
    SYSTEM.OUT.PRINTLN("BERNYANYI DENGAN IRAMA JAZZ");
  }
}

CLASS PENYANYIPOP EXTENDS PENYANYI {
  // MELAKUKAN OVERRIDE TERHADAP METHOD BERNYANYI()
  PUBLIC VOID BERNYANYI() {
    SYSTEM.OUT.PRINTLN("BERNYANYI DENGAN IRAMA POP");
  }
}
```

```
CLASS PENYANYIDANGDUT EXTENDS PENYANYI {
  // MELAKUKAN OVERRIDE TERHADAP METHOD BERNYANYI()
  PUBLIC VOID BERNYANYI() {
    SYSTEM.OUT.PRINTLN("BERNYANYI DENGAN IRAMA DANGDUT");
  }
}

CLASS DEMOPOLIMORFISME1 {
  PUBLIC STATIC VOID MAIN(STRING[] ARGS) {

    PENYANYI P;

    PENYANYIJAZZ MUSMUJIONO = NEW PENYANYIJAZZ();
    PENYANYIPOP AUDI = NEW PENYANYIPOP();
    PENYANYIDANGDUT INUL = NEW PENYANYIDANGDUT();

    P = MUSMUJIONO;  // P MENGACU PADA OBJEK PENYANYIJAZZ
    // AKAN MEMANGGIL METHOD PADA KELAS PENYANYIJAZZ
    P.BERNYANYI();

    P = AUDI;  // P MENGACU PADA OBJEK PENYANYIPOP
    // AKAN MEMANGGIL METHOD PADA KELAS PENYANYIPOP
    P.BERNYANYI();

    P = INUL;  // P MENGACU PADA OBJEK PENYANYIDANGDUT
    // AKAN MEMANGGIL METHOD PADA KELAS PENYANYIDANGDUT
    P.BERNYANYI();
  }
}
```

KELAS ABSTRAK

- PADA KASUS TERTENTU, KITA INGIN MENDEKLARASIKAN SEBUAH KELAS INDUK YANG MEMPUNYAI METHOD DIMANA METHOD TERSEBUT TIDAK MEMERLUKAN IMPLEMENTASI SAMA SEKALI.
- METHOD TERSEBUT BARU AKAN DIIMPLEMENTASI OLEH KELAS-KELAS TURUNANNYA. METHOD INI YANG DISEBUT **METHOD ABSTRAK**.
- METHOD ABSTRAK TIDAK DAPAT DIDEKLARASIKAN DENGAN TINGKAT *PRIVATE,* KARENA HARUS DIIMPLEMENTASI OLEH KELAS TURUNAN.
- BENTUK UMUM PENULISAN KODENYA:

```
ABTRACT TIPE NAMAMETHOD(DAFTAR-PARAMETER);
```

KELAS ABSTRAK

- SETIAP KELAS YANG DIDALAMNYA TERDAPAT SATU ATAU LEBIH METHOD ABSTRAK HARUS DIDEKLARASIKAN JUGA SEBAGAI KELAS ABSTRAK.
- BENTUK UMUM PENULISAN KODENYA:

```
ABTRACT CLASS NAMAKELAS{
//BADAN KELAS
}
```

KELAS ABSTRAK

- KELAS ABSTRAK TIDAK DAPAT DIINSTANTIASI (KITA TIDAK DIIZINKAN UNTUK MEMBENTUK OBJEK DARI SUATU KELAS ABSTRAK).
- MESKIPUN DEMIKIAN, KITA DIPERBOLEHKAN UNTUK MENDEKLARASIKAN SEBUAH VARIABEL REFERENSI KE KELAS ABSTRAK. SELANJUTNYA, VARIABEL REFERENSI TERSEBUT DAPAT DIGUNAKAN UNTUK MENGACU KE OBJEK-OBJEK DARI KELAS TURUNAN.
- CONTOH : DEMOKELASABSTRAK1.JAVA

KELAS ABSTRAK

CONTOH DEMOKELASABSTRAK1.JAVA

```
// MENDEFINISIKAN KELAS ABSTRACT
ABSTRACT CLASS A {

  // METHOD ABSTRAK, TIDAK MEMILIKI KODE IMPLEMENTASI
  ABSTRACT PUBLIC VOID COBA();

  // KELAS ABSTRAK MASING DIIZINKAN UNTUK MENDEFINISIKAN
  // METHOD NON-ABSTRAK
  PUBLIC VOID COBAJUGA() {
    SYSTEM.OUT.PRINTLN("METHOD NON-ABSTRAK " +
      "DI DALAM KELAS ABSTRAK");
  }
}

CLASS B EXTENDS A {

  // OVERRIDE METHOD COBA()
  PUBLIC VOID COBA() {
    SYSTEM.OUT.PRINTLN("METHOD DI DALAM KELAS B");
  }
}
```

```
CLASS DEMOKELASABSTRAK1 {
  PUBLIC STATIC VOID MAIN(STRING[] ARGS) {

    A REF; // MENDEKLARASIKAN REFERENSI KE KELAS A

    B OBJ = NEW B();

    REF = OBJ; // REF MENUNJUK KE OBJEK DARI KELAS B

    REF.COBA();     // MEMANGGIL METHOD COBA() DI DALAM B
    REF.COBAJUGA(); // MEMANGGIL METHOD COBAJUGA() DI DALAM A
  }
}
```

S E L E S A I