**MUHAMAD SHALMAN**

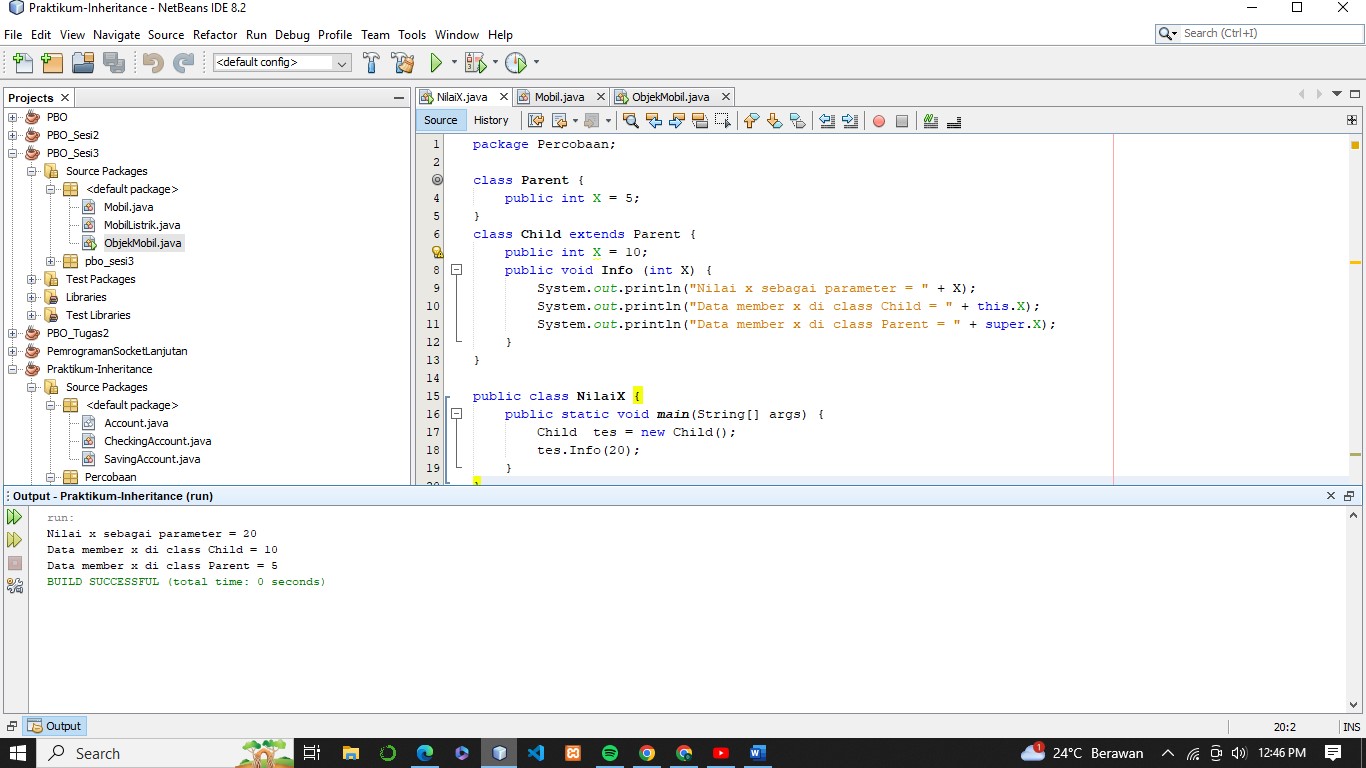
**20220040104**

**TI 22 J**

**PEMROGRAMAN BERORIENTASI OBJEK**

**Percobaan 1 :**

Percobaan ini menunjukan penggunaan kata kunci “super”.

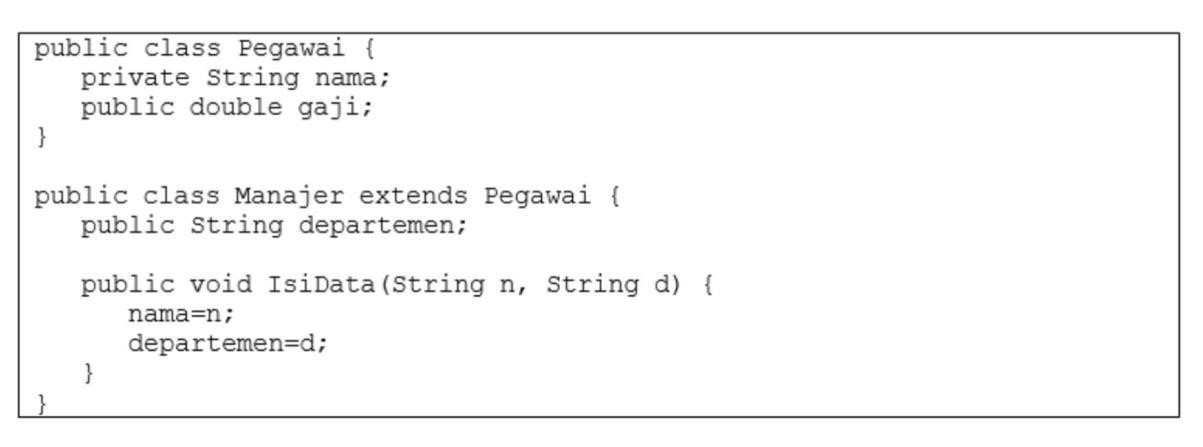


**Hasil Analisa :**

Pada percobaan 1 class` **Child**` mewariskan class Parent. Fungsi tes.info(20) akan menset nilai x menjadi 20 yang terdapat pada class `**Child**` dengan nama objek test. Hasil output akan menunjukan nilai x sebagai parameter = 20. Data member x di class `**Child**` = 10 karena di class `**Child**` sudah didefinisikan x menjadi 10. Lalu Data member x di class `**Parent**` = 5.

**Percobaan 2 :**

Percobaan berikut ini menunjukkan penggunaan kontrol akses terhadap atribut parent class.Mengapa terjadi error, dan bagaimana solusinya?



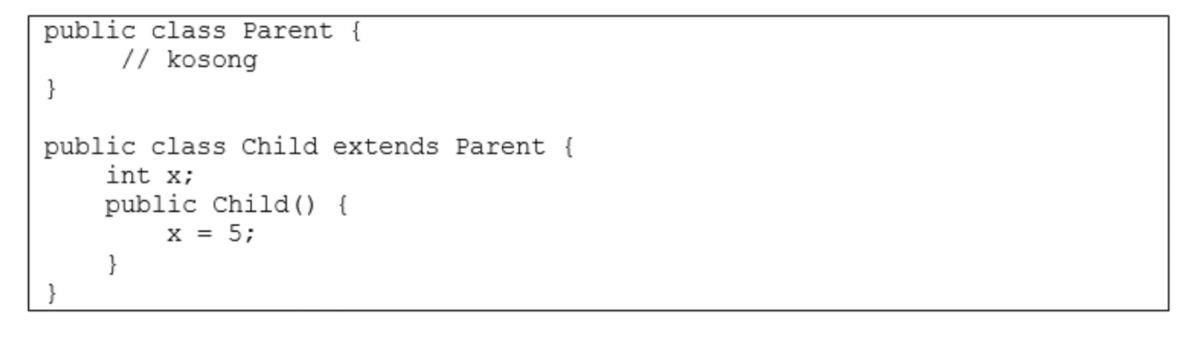
**Hasil Analisa :**

Error yang pertama terjadi karena class `**Manajer**` dinyatakan sebagai public seharusnya public digunakan ketika kelas `**Manajer**` diletakan pada filenya sendiri. Error kedua terjadi karena fungsi IsiData di class `**Manajer**` memanggil variabel nama dari class `**Pegawai**`, akan tetapi variabel nama diclass `**Pegawai`** di private sehingga terjadi error.

Solusinya adalah dengan mengubah modifier atribut nama pada class `**Pegawai`** menjadi public agar bisa diakses oleh class `**Manajer`**, lalu menghapus public pada class `**Manajer**` menjadi class `**Manajer**` saja.

**Percobaan 3 :**

Percobaan berikut ini menunjukkan penggunaan konstruktor yang tidak diwariskan. Mengapa

 terjadi error, dan bagaimana solusinya?

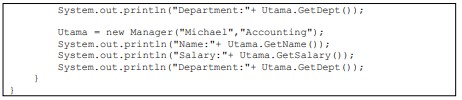
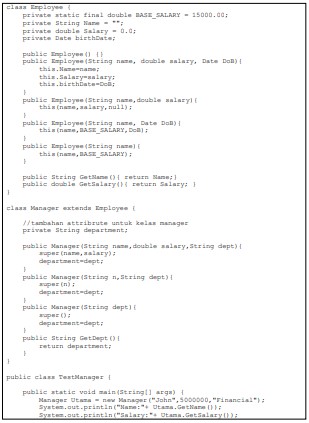
**Hasil Analisa :**

Percobaan tersebut menghasilkan error karena konstruktor di kelas `Child` mencoba mendefinisikan variabel lokal `x` dengan nilai 5, yang bukan merupakan inisialisasi variabel instance `x` dari kelas tersebut. Sebagai hasilnya, variabel instance `x` dari kelas `Child` tidak pernah diinisialisasi, yang dapat menyebabkan masalah ketika mencoba mengaksesnya.

Solusinya adalah dengan menghapus deklarasi variabel lokal **int x = 5;** di dalam konstruktor `Child**`**, sehingga konstruktor dapat menggunakan dan menginisialisasi variabel instance **x** yang diwarisi dari kelas `Parent`.

**Percobaan 4 :**

Percobaan berikut ini menunjukkan penggunaan kelas Employee dan subkelas Manager yang merupakan turunannya. Kelas TestManager digunakan untuk menguji kelas Manager.



**Hasil Analisa :**

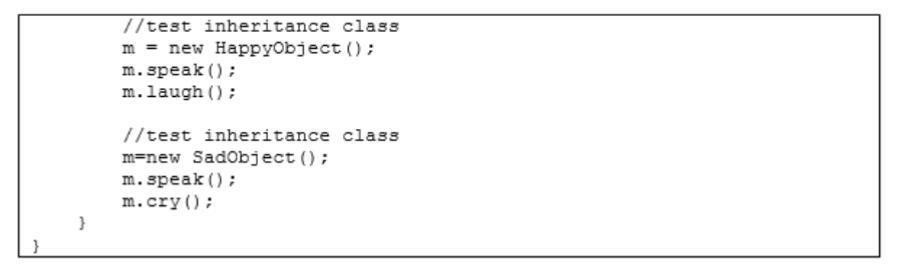
Program akan error bila tidak mengimport terlebih dahulu library **import java.util.Date;**, selebihnya kode program berjalan dengan lancar.

**Percobaan 5 :**

Percobaan berikut ini menunjukkan penggunaan kelas MoodyObject dengan subkelas HappyObject dan SadObject. Kelas MoodyTest digunakan untuk menguji kelas dan subkelas.

|  |  |
| --- | --- |
| − | SadObject berisi :  o sad, method untuk menampilkan pesan, tipe public |
| − | HappyObject berisi :  o laugh, method untuk menampilkan pesan, tipe public |
| − | MoodyObject berisi :  o getMood, memberi nilai mood sekarang, tipe public, return typestring o speak, menampilkan mood, tipe public |



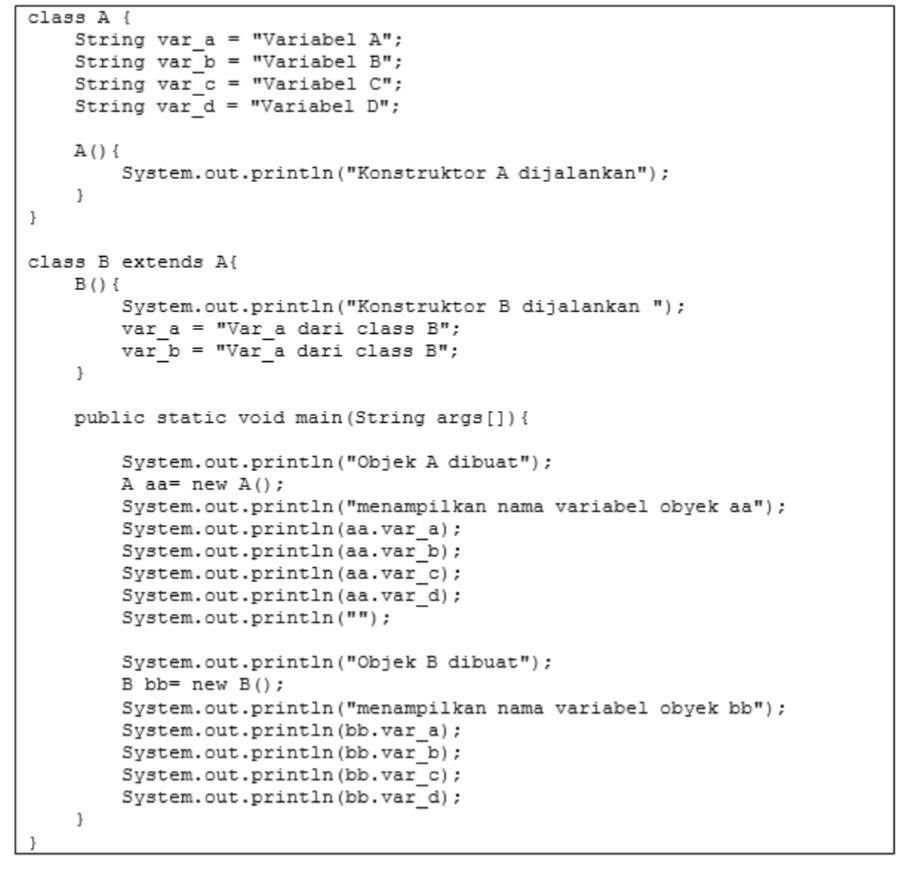


**Hasil Analisa :**

Terdapat error pada program dikarenakan setiap class dinyatakan public. Solusinya adalah menghapus setiap class yang dinyatakan public dan hanya menyisakan class **MoodyObject** dengan modifier public. Selebihnya program berjalan sesuai dengan fungsi fungsi yg telah dibuat.

**Percobaan 6 :**

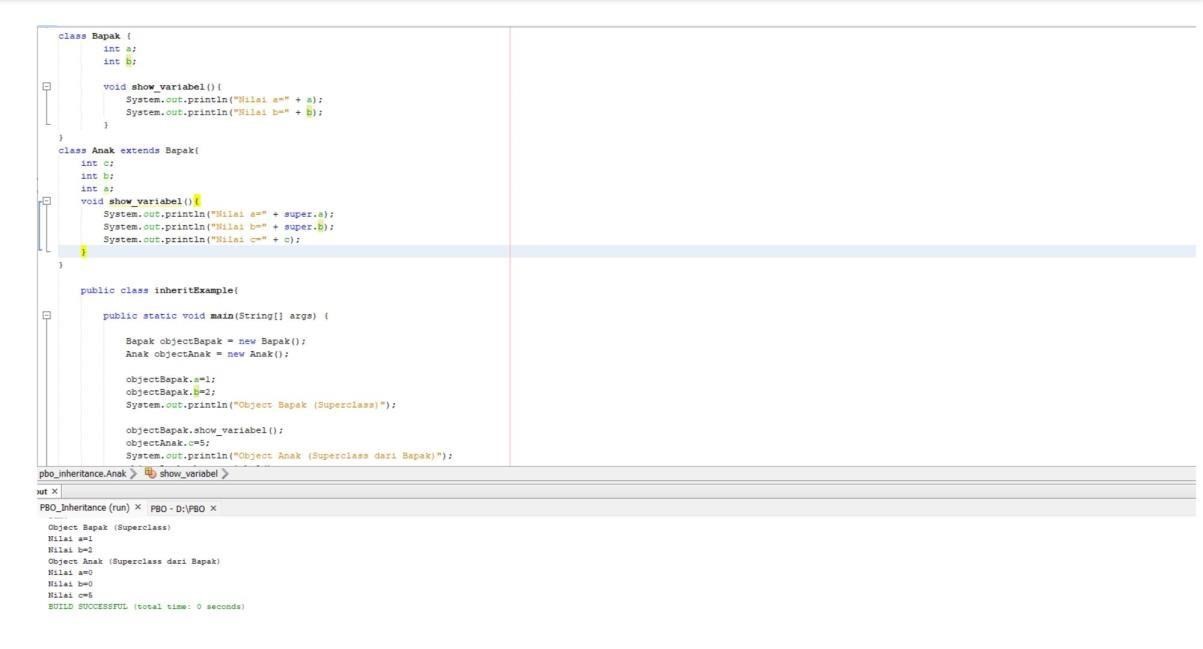
Percobaan berikut ini menunjukkan penggunaan kelas A dan dengan subkelas B. Simpan kedua kelas ini dalam 2 file yang berbeda (A.java dan B.java) dan dalam satu package. Perhatikan proses pemanggilan konstruktor dan pemanggilan variabel



**Hasil Analisa :**

Terdapat dua class yaitu kelas A sebagai parent dan class B sebagai subclass dari parent A.Pada percobaan ini class A dan class B dijalankan dalam file berbeda. Class B masih dapat mengakses kelas A karena pada dasarnya modifier default membuat class B dapat mengakses class A yang terdapat pada satu package yang sama.

**Percobaan 7 :**

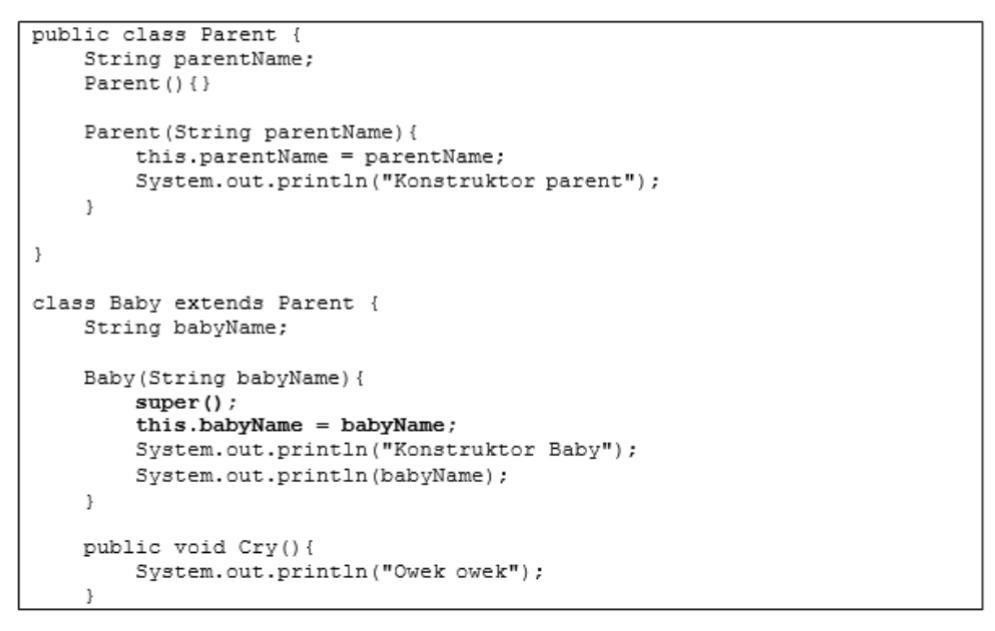


**Hasil Analisa :**

Walaupun sudah melakukan modifikasi pada method show\_variabel pada class anak dengan menggunakan super untuk menampilkan nilai a dan b nilainya akan tetap 0. Karena nilai dasarnya 0. Jadi, objek subclass tidak akan melakukan “Override” pada objek Bapak selama masih dalam bentuk objek.

**Percobaan 8 :**

Percobaan berikut ini menunjukkan penggunaan overriding method pada kelas Parent dan subkelas Baby, saat dilakukan pemanggilan konstruktor superclass dengan menggunakan super.



**Hasil Analisa :**

Pada kelas Parent menurunkan Baby. Terdapat super() pada fungsi constructor yang akan meng-override class parent-nya. this.babyName = babyName untuk melempar nilai babyName pada objek dengan parameter constructor babyName atau bisa juga Class Baby mewariskan class Parent. Pada class Baby terdapat super untuk meng-override class Parent nya. Atribut babyName diset pada constructor. Ketika class Baby dibuat akan menampilkan print dari constructor kelas Parent juga.