Fakultas Ilmu Terapan
School of Applied Science
Telkom University

MODUL PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

JUDUL: Kelas dan Objek

Dosen Pengampu:
Hanang Priambodo, S.T., M.Kom.
Pramuko Aji, S.T., M.T.
Furqon Hensan Muttaqien, S.Kom., M.Kom.
Maulida Mazaya, S.ST., Ph.D.

Daftar Isi

Daftar Kode Sumber

Daftar Gambar

Daftar Tabel

Panduan Penulisan Laporan Praktikum

0.1 Struktur dan Aturan Penulisan

Adapun struktur penulisan pada setiap *chapter* terdiri setidaknya yaitu *section* dan *subsection*. Adapun struktur heading pada setiap *chapter* (Kecuali *chapter assessment*):

1. **Tujuan Praktikum**.
   Tujuan praktikum pada buku laporan hasil dapat menyalin dari tujuan praktikum pada modul praktikum
2. **Dasar Teori**.
   Dasar Teori berisikan pernyataan referensi teori yang berkaitan dengan judul praktikum yang sedang dilaksanakan.

   Pada buku laporan praktikum dasar teori harus terdiri **minimal dua referensi** yang berbeda dari buku modul praktikum.
3. **Alat dan Bahan**.
   Section alat dan bahan berisikan kebutuhan untuk terimplementasinya praktikum, seperti: kebutuhan perangkat lunak dan perangkat keras.

   a) **Perangkat Lunak**.
   Subsection perangkat lunak berisikan spesifikasi perangkat lunak yang digunakan selama praktikum, seperti: sistem operasi, IDE, dan sebagainya

   b) **Perangkat Keras**.
   Subsection perangkat keras berisikan spesifikasi perangkat keras yang digunakan selama praktikum, seperti: besaran hardisk, memory, dan CPU.

4. **Langkah Kerja**.
   Section Langkah kerja berisikan langkah demi langkah dalam pembuatan program sesuai dengan topik praktikum yang sedang diselenggarakan. Adapun pada langkah kerja terdiri dari dua subsection yaitu persiapan dan implementasi Percobaan.
   a) **Persiapan Percobaan**.
      Subsection persiapan percobaan berisikan langkah-langkah awal dalam melaksanakan praktikum.
   b) **Implementasi Percobaan**.
      Subsection 4b (implementasi percobaan) berisikan langkah-langkah dan sumber kode yang harus ditulis, diterapkan serta *screenshoot* hasil program yang telah dijalankan.
5. **Hasil dan Pembahasan**.
   Setelah Implementasi program dilaksanakan, Mahasiwa membahas terkait bagaimana hasil dari implementasi, seperti:
   - Penjelasan maskud dari program yang diimplementasikan.
   - Penjelasan setiap bagian kode sumber.
   - Hasil keluaran program yang diimplementasikan.
6. **Kesimpulan**.
   Section kesimpulan berisikan simpulan pernyataan dari hasil yang telah dibahas.
7. **Tugas Akhir**. Soal tugas akhir pada setiap pertemuan disalin pada bagian ini.
   a) **Implementasi Program Tugas**.
      subsection ini berisikan langkah-langkah dan sumber kode pada Tugas Akhir, serta *screenshoot* hasil program yang telah dijalankan. Penulisan program pada buku ini harus mengikuti format perintah pada *subsection* 0.2.8.
   b) **Pembahasan Hasil Tugas**.
      Subsection ini berisikan pernyataan terkait pembahasan hasil implemantasi Program tugas akhir.
8. **Referensi**.
   Berisikan daftar referensi yang ada pada dasar teori.

Pada *chapter* assessment tidak diperlukan *section* dasar teori, tugas akhir, dan referensi.

0.2 Standar Penulisan pada Latex

Pembuatan buku laporan praktikum Pemrograman Berorientasi Objek diwajibkan untuk menggunakan editor LaTeX. Penggunaan LaTeX akan memastikan format yang konsisten dan profesional, serta memudahkan dalam pengelolaan referensi dan struktur dokumen.

0.2.1 Pengoprasian Latex

Untuk memudahkan mahasiswa dalam penulisan buku menggunakan LaTeX, mahasiswa disarankan untuk memanfaatkan penyunting LaTeX online, yaitu `https://www.overleaf.com/`. Dokumentasi lengkap mengenai penggunaan LaTeX dapat diakses melalui URL berikut: `https://www.overleaf.com/learn`.

Adapun langkah awal dalam penggunaan overleaf, yaitu:

1. Akses website overleaf
2. Jika mahasiswa belum mempunyai akun overleaf, lakukan pendaftaran pada menu "`Sign up`".
3. Jika mahasiswa telah mendaftar, pilih "`Log in`" dan masukan akun yang telah terdaftar.
4. Setelah mahasiswa berhasil `Log in`, mahasiswa membuat proyek baru menggunakan menu "`new project`". Kemudian pilih menu "`upload project`".
5. Mahasiswa mengunggah template_laporan_praktikum_pbo.zip. Berkas template laporan dapat didownload pada url:

   `https://tel-u.ac.id/templatepraktikumpbo`
   atau
   `https://teamgit.telkomuniversity.ac.id/hanangpriambodo/pbo-gck1mab4/-/blob/main/template-buku-laporan-praktikum/Template-laporan-praktikum-PBO.zip?ref_type=heads`

Mahasiswa hanya perlu fokus pada direktori *chapters* dan *images*. Direktori *chapters* berisi 15 berkas. Setiap berkas diawali dengan urutan nomor, di mana setiap angka pada berkas mencerminkan minggu pertemuan praktikum atau *chapters* dalam buku.

Pastikan bahwa setiap penulisan laporan praktikum sesuai dengan judul *chapter* praktikum yang telah disediakan dalam template. Adapun judul praktikum yaitu:

1. Instalasi Software
2. Pengenalan Java #1
3. Pengenalan Java #2
4. Kelas dan objek
5. assessment 1
6. Enkapsulasi
7. Pewarisan dan polimorfisme
8. Kelas Abstrak dan Antarmuka
9. Relasi antar kelas
10. Assessment 2
11. Scene dan Stage pada JavaFX
12. Layout dan KontrolUI pada JavaFX
13. Penanganan *Event* pada JavaFX
14. CRUD dan Database pada JavaFX
15. Tugas Besar dan Assessment 3

Pada direktori images terdapat 15 direktori yang diawali dengan kata `chapter\_`. Angka disetiap penamaan direktori *chapter* menunjukkan minggu pertemuan atau judul praktikum seperti angka pada penaman berkas yang ada di direktori *chapters*. Setiap direktori `images/ chapter_{angka}` berisikan gambar yang diunggah.

Terdapat empat berkas diluar direktori `chapters`, `images` ataupun `lib`, yaitu:

- **cover_page.tex.** Berisikan sintaks untuk menyunting sampul.

- **ebookebook.tex.** Berkas utama dari buku laporan. berisikan uratan *chapter* yang akan ditampilkan pada buku.
- **info_praktikan.tex.** Berkas berisikan sintaks untuk menulis informasi praktikan, seperti: nama, kelas, nim dan dosen pengampu.
- **referensi.bib.** Berisikan informasi literatur atau referensi.

Penulisan Informasi Praktikan

Praktikan wajib menuliskan informasi seperti: nama lengkap praktikan, kelas, nip, dan dosen pengampu pada berkas `info_praktikan.tex`. Sintaks yang ada didalan berkas `info_praktikan.tex`, yaitu:

```
\newcommand{\namalengkappraktikan}{Masukan Nama Lengkap}
\newcommand{\nimpraktikan}{Masukan NIM}
\newcommand{\kelaspraktikan}{Masukan kelas}
\newcommand{\dosenpengampu}{Masukan Nama Dosen Pengampu}
```

Pada bagian `Masukan Nama Lengkap` ganti informasi tersebut menjadi nama lengkap praktikan. Selanjutnya, Pada bagian `Masukan NIM` ganti informasi tersebut menjadi NIM praktikan. Selanjutnya, Pada bagian `Masukan Kelas` ganti informasi tersebut menjadi kelas praktikan. Kemudian yang terakhir, ganti informasi pada bagian `Masukan Nama Dosen Pengampu` menjadi Nama Dosen yang mengajar PBO pada kelas praktikan saat ini.

0.2.2 Penulisan Referensi

Setiap pernyataan yang disampaikan pada dasar teori harus diakhiri dengan kode acuan referensi yang relevan.

Untuk menambahkan referensi, dapat digunakan format BibTeX, sebagaimana ditunjukkan dalam Kode Sumber 1.

```
@book{liang2015introduction,
title={Introduction to Java programming},
author={Liang, Y Daniel},
year={2015},
publisher={Pearson Boston},
```

```
keywords={panduan_penulisan}
}
```

Kode Sumber 1: Penambahan Referensi

Setiap informasi pada satu referensi harus dipisahkan dengan tanda koma (,). Pastikan setiap penambahan referensi seperti pada kode sumber 1 kedalam berkas `referensi.bib`, tambahkan baris `keywords` seperti pada baris keenam dari Sumber Kode 1. Baris keywords ini berfungsi agar referensi dapat ditampilkan sesuai dengan bab-nya (*chatper*). Adapun keywords pada setiap *chapter* (Praktikum) yaitu:

- Panduan penulisan: `keywords=panduan_penulisan`
- Instalasi software: `keywords=instalasi_software`
- Pengenalan Java #1: `keywords=pengenalan_java_1`
- Pengenalan Java #2: `keywords=pengenalan_java_2`
- Kelas dan objek: `keywords=kelas_dan_objek`
- Assessment 1: `keywords=assessment_1`
- Enkapsulasi: `keywords=enkapsulasi`
- Pewarisan dan Polimorfisme: `keywords=pewarisan_dan_polimorfisme`
- Kelas abstrak dan antarmuka: `keywords=kelas_abstrak_dan_antarmuka`
- Relasi antar kelas: `keywords=relasi_antar_kelas`
- Assessment 2: `keywords=assessment_2`
- Framework JavaFX: `keywords=framework_javafx`
- Layout dan KontrolUI: `keywords=layout_dan_kontrolui`
- Penanganan Event: `keywords=penanganan_event`
- CRUD dan database: `keywords=crud_dan_database`
- Tugas Besar dan Assessment 3: `keywords=tugas_besar_dan_assessment_3`

Selanjutnya, untuk mengintegrasikan referensi yang telah disimpan ke dalam suatu paragraf, dapat digunakan perintah `\cite{}`, seperti yang diperlihatkan dalam Kode Sumber 2.

```
\cite{liang2015introduction}
```

Kode Sumber 2: Perintah Penggunaan Referensi

Sebagai contoh sebuah pernyataan ditulis seperti:

Java adalah bahasa pemrograman yang dikembangkan oleh Sun Microsystems, yang sekarang merupakan bagian dari Oracle Corporation [1].

Pada contoh diatas, maka pada "Code Editor" latex dapat dituliskan:

```
Java adalah bahasa pemrograman yang dikembangkan oleh Sun
   Microsystems, yang sekarang merupakan bagian dari Oracle
   Corporation \cite{liang2015introduction}.
```

Kode Sumber 3: Contoh Penulisan Referensi

0.2.3 Penggunaan gaya tulisan

Pada mudul praktikum ini menggunakan dua grafis karakter, yaitu:

1. Tipe normal.
   Penggunaan tipe normal seperti yang digunakan saat ini.
2. Tipe monospace.
   Penggunaan tulisan dengan gaya monospace, dapat menggugunakan perintah `\texttt{}`. Berikut adalah contoh penggunaan monospace: `\texttt{monospace}`, hasilnya adalah `monospace`. Tipe monospace pada buku laporan digunakan untuk penulisan, seperti:
   - **Sintaks** yang berada di paragraf.
   - **Path** (Lokasi suatu file atau direktori).
   - **Nama Berkas java**.
3. Tipe Tebal (**BOLD**).
   Pada latex menuliskan kata menjadi tebal dapat menggunakan perintah `\textbf{}`. Sebagai contoh: `\textbf{Tebal}` menghasilkan bentuk kata **Tebal**.
4. Tipe Miring.
   Pada latex menuliskan kata menjadi miring dapat menggunakan perintah `\textit{}`. Sebagai contoh: `\textit{Miring}` menghasilkan bentuk kata *Miring*.

0.2.4 Pergantian paragraf

. Pada editor latex pergantian paragraf dapat dilakukan dengan 2 cara, yaitu:

- Menambahkan baris kosong diantara paragraf.

```
1 Paragraf 1
2
3 Paragraf 2
```

Kode Sumber 4: Pergantian Paragraf (Contoh 1)

- Menggunakan instruksi `\par`.

```
1 Paragraf 1
2 \par Paragraf 2
```

Kode Sumber 5: Pergantian Paragraf (Contoh 2)

0.2.5 Pengaturan hyphenation

Hyphenation adalah teknik dalam penulisan yang berfungsi untuk pemenggalan sebuah kata agar dapat penyesuaikan kolom pada teks. Misalkan terdapat kalimat:

> `Ini merupakan contoh dari kalimat dimana kata yang berada dibagian kanan telah melewati batas.`

Dari contoh diatas terlihat kata `dibagian` telah melewati batas dari sebelah kanan. Untuk melakukan pemenggalan dapat dilakukan dua cara, yaitu:

1. Langsung melakukan pemenggalan pada paragraf dengan cara menuliskan kata "`dibagian`" menjadi "`di\-bagi\-an`", atau
2. Menuliskan setiap pemenggalan kata pada berkas `_internals/hype.indonesia.tex`. Cara penulisannya adalah menggunakan tanda −. Contoh, kata "`dibagian`", pada berkas `_internals/hype.indonesia.tex` ditulis menjadi `di-bagi-an`.

Hasil dari penggunaan pemenggalan dari contoh diatas adalah:

> `Ini merupakan contoh dari kalimat dimana kata yang berada dibagian kanan telah melewati batas.`

0.2.6 Penambahan Gambar

Latex memungkinkan untuk menampilkan suatu gambar dengan menuliskan perintah:

```
1 \begin{figure}[H]
2     \centering
3     \includegraphics[width=0.5\linewidth]{path_gambar}
4     \caption{Caption Gambar}
5     \label{fig:laber-gambar}
6 \end{figure}
```

Kode Sumber 6: Sintaks Menggunakan Gambar

Gambar 1: Gambar Univesitas Telkom

Pada sintaks `\includegraphics[width=0.5\linewidth]{path_gambar}` baris ketiga dari Kode Sumber 6, berfungsi untuk memasukkan gambar. Mahasiswa hanya perlu merubah kata `path_gambar` menjadi path yang sebenranya, seperti: `images/buku/Logo_Telkom_University_ potrait.png`. Kemudian, untuk mengatur size suatu gambar, mahasiswa hanya perlu untuk mengatur pada bagian width dibaris ketida pada Kode Sumber 6.

0.2.7 Penambahan Tabel

Penulisan tabel pada latex dapat menggunakan sintaks:

```
\begin{table}[H]
    \centering
    \begin{tabular}{|c|c|}\hline
        contoh & contoh \\ \hline
        data1 & data2 \\ \hline
    \end{tabular}
    \caption{Contoh Tabel}
    \label{tab:my_label}
\end{table}
```

Kode Sumber 7: Contoh Pembuatan Tabel

Adapun untuk menambah kolom dapat menambahkan karakter "c" kemudian ditutup dengan simbol | pada baris ketiga dari Kode Sumber 7 kemudian setiap baris pada tabel data perlu ditambahkan "& data" sebelum tanda "\\ \hline". kemudian untuk menambah baris, lakukan seperti dibaris keempat dan kelima.

contoh	contoh
data1	data2

Tabel 1: Contoh Tabel

0.2.8 Penulisan Pencatatan Kode

Dalam penulisan kode sumber pada latex dapat ditulikan menggunakan perintah `lstlisting`. Adapun contoh sintaks tersebut dapat dilihat pada Kode Sumbet 8.

```
\begin{lstlisting}[language=Java, caption=Nama Caption, label=
   sc:namaLabelPencatatanKode]
class Contoh(){
    //...Tulis program disini
}
\end{lstlisting }
```

Kode Sumber 8: Contoh Penulisan Kode Program

Hasil dari sintaks Kode Sumber 8 dapat dilihat pada Kode Sumber 9

```
class Contoh(){
    //...Tulis program disini
}
```

Kode Sumber 9: Hasil Contoh Pencatatan Kode

Dalam melakukan penulisan kode progam, pemberian nama caption dan label dapat dilakukan pada baris pertama seperti pada Kode Sumber 8. Caption adalah nama yang tertulis pada buku merujuk pada penomoran kode sumber. Sedangkan label adalah parameter yang dapat dipanggil menggunakan sintaks `\ref{}`. Penamaan `label` sebaiknya tidak terdapat whitespace (spasi). Pada buku ini untuk format penulisan `label` kode sumber adalah `sc:nama-label` (Pemberian label selalu didahului dengan kode `sc:`).

0.2.9 Penulisan *bullet* dan *numbering*

Terdapat dua tipe dalam penulisan daftar, pertama adalah tipe *numberring* (`enumerate`) dimana daftar diurutkan bedasarkan nomor, dan tipe *bullet* (`itemize`).

Dalam penulisan *numbering* seperti dibawah ini:

1. daftar A
2. daftar B

dapat menggunakan perintah seperti pada Kode Sumber 10

```
\begin{enumerate}
    \item daftar A
    \item daftar B
\end{enumerate}
```

Kode Sumber 10: Sintaks Penulisan Penomoran

Penulisan daftar dalam bentuk symbol terlihat seperti berikut:

- Daftar 1
- Daftar 2

Sintaks yang dapat digunakan adalah `itemize`, seperti pada kode sumber 11):

```
\begin{itemize}
    \item daftar 1
    \item daftar 2
\end{itemize}
```

Kode Sumber 11: Sintaks Penulisan Daftar Simbol

Referensi

[1] Y Daniel Liang. *Introduction to Java programming*. Pearson Boston, 2015.

1 Pewarisan dan polimorfisme

Pewarisan dan polimorfisme adalah dua konsep fundamental dalam pemrograman berorientasi objek (PBO) yang sangat penting untuk dipahami oleh setiap pengembang perangkat lunak. Pewarisan memungkinkan pengembang untuk menciptakan kelas baru yang mewarisi atribut dan metode dari kelas yang sudah ada, sehingga memfasilitasi penggunaan kembali kode dan pengorganisasian hierarki kelas. Dengan pewarisan, pengembang dapat mengurangi duplikasi kode dan meningkatkan efisiensi dalam pengembangan perangkat lunak.

Sementara itu, polimorfisme memberikan fleksibilitas dalam penggunaan objek, memungkinkan satu antarmuka untuk digunakan dengan berbagai tipe objek. Konsep ini memungkinkan metode yang sama untuk berfungsi dengan cara yang berbeda tergantung pada objek yang memanggilnya. Dengan demikian, polimorfisme mendukung prinsip desain yang lebih baik dan meningkatkan kemampuan pemeliharaan kode.

Melalui modul praktikum ini, mahasiswa akan diajak untuk memahami dan menerapkan konsep pewarisan dan polimorfisme dalam pemrograman Java. Praktikum ini dirancang untuk memberikan pengalaman langsung dalam mengimplementasikan kedua konsep tersebut, sehingga mahasiswa dapat mengembangkan keterampilan yang diperlukan untuk menciptakan aplikasi yang lebih terstruktur dan efisien. Diharapkan, pemahaman yang mendalam tentang pewarisan dan polimorfisme akan membantu mahasiswa dalam menghadapi tantangan dalam pengembangan perangkat lunak di masa depan.

1.1 Tujuan Praktikum

Praktikum ini bertujuan untuk memberikan pemahaman yang mendalam tentang konsep pewarisan dan polimorfisme dalam pemrograman Java. Pewarisan memungkinkan pengembang untuk menciptakan kelas baru yang mewarisi atribut dan metode dari kelas yang sudah ada, sehingga memfasilitasi penggunaan kembali kode dan pengorganisasian hierarki kelas. Sementara itu, polimorfisme memungkinkan objek untuk diperlakukan sebagai instansi dari kelas induk mereka, sehingga metode yang sama dapat berfungsi dengan cara yang berbeda tergantung pada objek yang memanggilnya. Melalui praktikum ini, mahasiswa diharapkan dapat mengimplementasikan kedua konsep tersebut dalam program nyata. Secara khusus, tujuan praktikum ini mencakup:

1. Memahami dan menjelaskan konsep dasar pewarisan dalam pemrograman Java.
2. Mampu menerapkan pewarisan untuk menciptakan hierarki kelas yang efisien.
3. Mengerti dan menerapkan polimorfisme dalam konteks metode dan objek.
4. Mampu mengintegrasikan pewarisan dan polimorfisme dalam implementasi program yang robust.

1.2 Dasar Teori

Pewarisan adalah mekanisme dalam pemrograman berorientasi objek yang memungkinkan sebuah kelas (kelas anak atau subclass) untuk mewarisi atribut dan metode dari kelas lain (kelas induk atau superclass). Dengan menggunakan pewarisan, pengembang dapat menciptakan hierarki kelas yang lebih terstruktur dan memudahkan penggunaan kembali kode. Misalnya, jika ada kelas Hewan, kelas Kucing dan Anjing dapat mewarisi atribut dan metode dari kelas Hewan, sehingga tidak perlu mendefinisikan ulang atribut yang sama.

Polimorfisme, di sisi lain, adalah kemampuan objek untuk mengambil banyak bentuk. Dalam Java, polimorfisme dapat dicapai melalui metode overriding dan interface. Dengan polimorfisme, metode yang sama dapat berfungsi dengan cara yang berbeda tergantung pada objek yang memanggilnya. Misalnya, metode `suara()` dapat diimplementasikan berbeda dalam `kelas` Kucing dan Anjing, meskipun keduanya merupakan subclass dari kelas Hewan. Konsep ini sangat penting dalam menciptakan kode yang fleksibel dan mudah dipelihara

1.3 Alat dan Bahan

Alat dan bahan yang digunakan dalam praktikum ini terdiri dari dua kategori utama, yaitu perangkat keras dan perangkat lunak. Perangkat keras mencakup semua komponen fisik yang diperlukan untuk menjalankan program, sedangkan perangkat lunak mencakup aplikasi dan sistem yang digunakan untuk menulis, menguji, dan menjalankan kode program.

1.3.1 Perangkat Lunak

- Integrated Development Environment (IDE): Visual Studio Code (Yang telah diinstal dan dikonfigurasi pada praktikum minggu pertama sebagai editor Java)

1.3.2 Perangkat Keras

Komputer atau laptop dengan spesifikasi minimum:

- Prosesor: Intel Core i3 atau setara
- RAM: 4 GB
- Ruang penyimpanan: 10 GB

1.4 Langkah Kerja

Pelaksanaan praktikum ini terdiri dari dua tahapan utama, yaitu persiapan dan implementasi program Java, yang dirancang secara sistematis untuk memastikan mahasiswa dapat mengikuti setiap langkah dengan baik serta mencapai tujuan praktikum secara optimal. Penjelasan lebih rinci mengenai kedua tahapan tersebut diuraikan pada *subsection* 1.4.1 persiapan dan 1.4.2 implementasi program.

1.4.1 Persiapan

Sebelum memulai praktikum, terdapat beberapa langkah persiapan yang perlu dilakukan guna memastikan kelancaran pelaksanaan kegiatan. Persiapan ini bertujuan untuk menyiapkan lingkungan pengembangan yang sesuai serta memastikan bahwa semua komponen yang diperlukan telah tersedia. Adapun langkah-langkah persiapan yang harus dilakukan adalah sebagai berikut:

1. Buka Visual Studio Code
2. Buat proyek java baru
3. Pilih tipe proyek `"No build tools"`
4. Kemudian berikan judul `pewarisan_dan_polimofisme`
5. Hapus berkas `App.java` pada folder `src`
6. Buat berkas java didalam folder `src` yang terdiri dari:
    - `HewanPewarisan.java`
    - `KucingPewarisan.java`
    - `HewanPolimorfisme.java`
    - `KucingPolimorfisme.java`
    - `SapiPolimorfisme.java`
    - `MainPewarisan.java`
    - `MainPolimorfisme.java`

1.4.2 Implementasi Percobaan

Setelah tahap persiapan selesai, langkah selanjutnya adalah mengimplementasikan program Java sesuai dengan konsep yang dipelajari.

Implementasi Pewarisan

Percobaan pertama yang dilakukan adalah percobaan penggunaan pustaka. Adapun langkah yang dilakukan adalah:

1. Buka berkas `HewanPewarisan.java` di visual studio code.
2. Masukan Kode Sumber 1.1 (Implementasi Pewarisan *Parrent*) ke berkas `HewanPewarisan.java`.

```java
// Kelas Induk (Superclass)
public class HewanPewarisan {
    protected String nama;

    // Konstruktor
    public HewanPewarisan(String nama) {
        this.nama = nama;
    }

    // Metode
    public void bergerak() {
        System.out.println(nama + " bergerak.");
    }
}
```

Kode Sumber 1.1: Implementasi Pewarisan (Parrent)

3. Simpan berkas `HewanPewarisan.java`
4. Buka berkas `KucingPewarisan.java` di visual studio code.
5. Masukan Kode Sumber 1.2 (Implementasi Pewarisan Turunan) ke berkas `KucingPewarisan.java`.

```java
// Kelas Turunan (Subclass)
public class KucingPewarisan extends HewanPewarisan {
    // Konstruktor
    public KucingPewarisan(String nama) {
        super(nama);
    }
```

```
7
8       // Metode overriding
9       @Override
10      public void bergerak() {
11          System.out.println(nama + " berlari dengan cepat."
        );
12      }
13
14      // Metode tambahan
15      public void bersuara() {
16          System.out.println(nama + " mengeluarkan suara
        meong.");
17      }
18  }
```

Kode Sumber 1.2: Implementasi Pewarisan (Turunan)

6. Simpan berkas `KucingPewarisan.java`.
7. Buka berkas `MainPewarisan.java` di visual studio code.
8. Masukan Kode Sumber 1.3 (Implementasi Main Pewarisan) ke berkas `MainPewarisan.java`.

```
1  public class MainPewarisan {
2      public static void main(String[] args) {
3          HewanPewarisan hewan = new HewanPewarisan("Hewan")
       ;
4          hewan.bergerak(); // Output: Hewan bergerak.
5
6          KucingPewarisan kucing = new KucingPewarisan("
       Kucing");
7          kucing.bergerak(); // Output: Kucing berlari
       dengan cepat.
8          kucing.bersuara(); // Output: Kucing mengeluarkan
       suara meong.
9      }
10 }
```

Kode Sumber 1.3: Implementasi Pewarisan (Main)

9. Simpan berkas `KucingPewarisan.java`.
10. Jalankan berkas `MainPewarisan.java`.
11. Simpan atau screenshoot hasilnya untuk dilaporkan pada buku laporan praktikum.

Percobaan pertama yang dilakukan adalah percobaan penggunaan pustaka. Adapun langkah yang dilakukan adalah:

1. Buka berkas `HewanPolimorfisme.java` di visual studio code.
2. Masukan Kode Sumber 1.4 (Implementasi Pewarisan *Parrent*) ke berkas `HewanPolimorfisme.java`.

```
// Kelas Induk (Superclass)
public class HewanPolimorfisme {
    protected String nama;

    // Konstruktor
    public HewanPolimorfisme(String nama) {
        this.nama = nama;
    }

    // Metode
    public void bergerak() {
        System.out.println(nama + " bergerak.");
    }
}
```

Kode Sumber 1.4: Implementasi Polimorfisme (Parrent)

3. Simpan berkas `HewanPolimorfisme.java`
4. Buka berkas `KucingPolimorfisme.java` di visual studio code.
5. Masukan Kode Sumber 1.5 (Implementasi Pewarisan Turunan) ke berkas `KucingPolimorfisme.java`.

```
// Kelas Turunan (Subclass)
public class KucingPewarisan extends HewanPolimorfisme {
    // Konstruktor
    public KucingPolimorfisme(String nama) {
        super(nama);
    }

    // Metode overriding
    @Override
    public void bergerak() {
        System.out.println(nama + " berlari dengan cepat."
    );
    }
```

```
13
14      // Metode tambahan
15      public void bersuara() {
16          System.out.println(nama + " mengeluarkan suara
        meong.");
17      }
18  }
```

Kode Sumber 1.5: Implementasi Polimorfisme (Turunan)

6. Simpan berkas `KucingPolimorfisme.java`.
7. Buka berkas `SapiPolimorfisme.java` di visual studio code.
8. Masukan Kode Sumber 1.6 (Implementasi Pewarisan Turunan) ke berkas `SapiPolimorfisme.java`.

```
1  // Kelas Turunan (Subclass)
2  public class SapiPolimorfisme extends HewanPolimorfisme {
3      // Konstruktor
4      public SapiPolimorfisme(String nama) {
5          super(nama);
6      }
7
8      // Metode overriding
9      @Override
10     public void bergerak() {
11         System.out.println(nama + " berlari dengan cepat."
       );
12     }
13
14     // Metode tambahan
15     public void bersuara() {
16         System.out.println(nama + " mengeluarkan suara
       moow.");
17     }
18 }
```

Kode Sumber 1.6: Implementasi Polimorfisme (Turunan)

9. Simpan berkas `SapiPolimorfisme.java`.
10. Buka berkas `MainPolimorfisme.java` di visual studio code.
11. Masukan Kode Sumber 1.7 (Implementasi Main Pewarisan) ke berkas `MainPolimorfisme.java`.

```java
public class MainPolimorfisme {
    public static void main(String[] args) {
        Hewan hewan1 = new Kucing("Kucing");
        Hewan hewan2 = new Anjing("Anjing");

        hewan1.bergerak(); // Output: Kucing berlari dengan cepat.
        hewan2.bergerak(); // Output: Anjing berlari dengan kuat.

        // Polimorfisme dinamis
        Hewan[] hewans = {hewan1, hewan2};
        for (Hewan hewan : hewans) {
            hewan.bergerak();
        }
        // Output:
        // Kucing berlari dengan cepat.
        // Anjing berlari dengan kuat.
    }
}
```

Kode Sumber 1.7: Implementasi Polimorfisme (Main)

12. Simpan berkas `KucingPewarisan.java`.
13. Jalankan berkas `MainPewarisan.java`.
14. Simpan atau screenshoot hasilnya untuk dilaporkan pada buku laporan praktikum.

1.5 Tugas Akhir

Buatlah kelas Transportasi sebagai kelas induk, dengan *subclass* `Mobil` dan `Sepeda`. Implementasikan metode `bergerak()` yang memberikan output berbeda untuk masing-masing *subclass*.

Tugas dilaporkan pada buku laporan praktikum pada section Tugas Akhir yang terdiri dari:

- Implementasi program, dan
- Pembahasan hasil.

Download laporan yang telah dibuat pada editor latex (overleaf) menjadi berkas PDF. Kemudian ubah nama laporan menjadi:

PBO-D3SI-{kelas}-Nama_Lengkap-NIM-{Judul_Praktikum}.pdf. Kumpulkan berkas PDF di **LMS**.

CATATAN PENTING:

Sebelum melakukan download PDF, pada berkas `ebookebook.tex` baris ke 46 sampai 50, ubah sintaks dari

```
\input{chapters/1_instalasi_software}
\input{chapters/2_pengenalan_java_#1}
\input{chapters/3_pengenalan_java_#2}
\input{chapters/4_kelas_dan_objek}
\input{chapters/5_assessment_1}
\input{chapters/6_enkapsulasi}
\input{chapters/7_pewarisan_dan_polimorfisme}
\input{chapters/8_kelas_abstrak_dan_antarmuka}
\input{chapters/9_relasi_antar_kelas}
\input{chapters/10_assessment_2}
\input{chapters/11_framework_javafx}
\input{chapters/12_layout}
\input{chapters/13_kontrolui}
\input{chapters/14_crud_dan_basisdata}
\input{chapters/15_tugas_besar_dan_assessment_3}
```

menjadi

```
% \input{chapters/1_instalasi_software}
% \input{chapters/2_pengenalan_java_#1}
% \input{chapters/3_pengenalan_java_#2}
% \input{chapters/4_kelas_dan_objek}
% \input{chapters/5_assessment_1}
% \input{chapters/6_enkapsulasi}
\input{chapters/7_pewarisan_dan_polimorfisme}
% \input{chapters/8_kelas_abstrak_dan_antarmuka}
% \input{chapters/9_relasi_antar_kelas}
% \input{chapters/10_assessment_2}
% \input{chapters/11_framework_javafx}
% \input{chapters/12_layout}
% \input{chapters/13_kontrolui}
% \input{chapters/14_crud_dan_basisdata}
% \input{chapters/15_tugas_besar_dan_assessment_3}
```

agar yang dicetak hanyalah chapter yang dilakukan praktikum saat ini.

Referensi

[1] Y Daniel Liang. *Introduction to Java programming*. Pearson Boston, 2015.