

LAPORAN PRAKTIKUM
TEKNIK PEMROGRAMAN
INHERITANCE, ABSTRACT CLASS, INTERFACE, POLYMORPHISM
MINGGU KE-4



POLBAN

NAMA: FAUZI ISMAIL

NIM: 241524042

KELAS: D4-1B

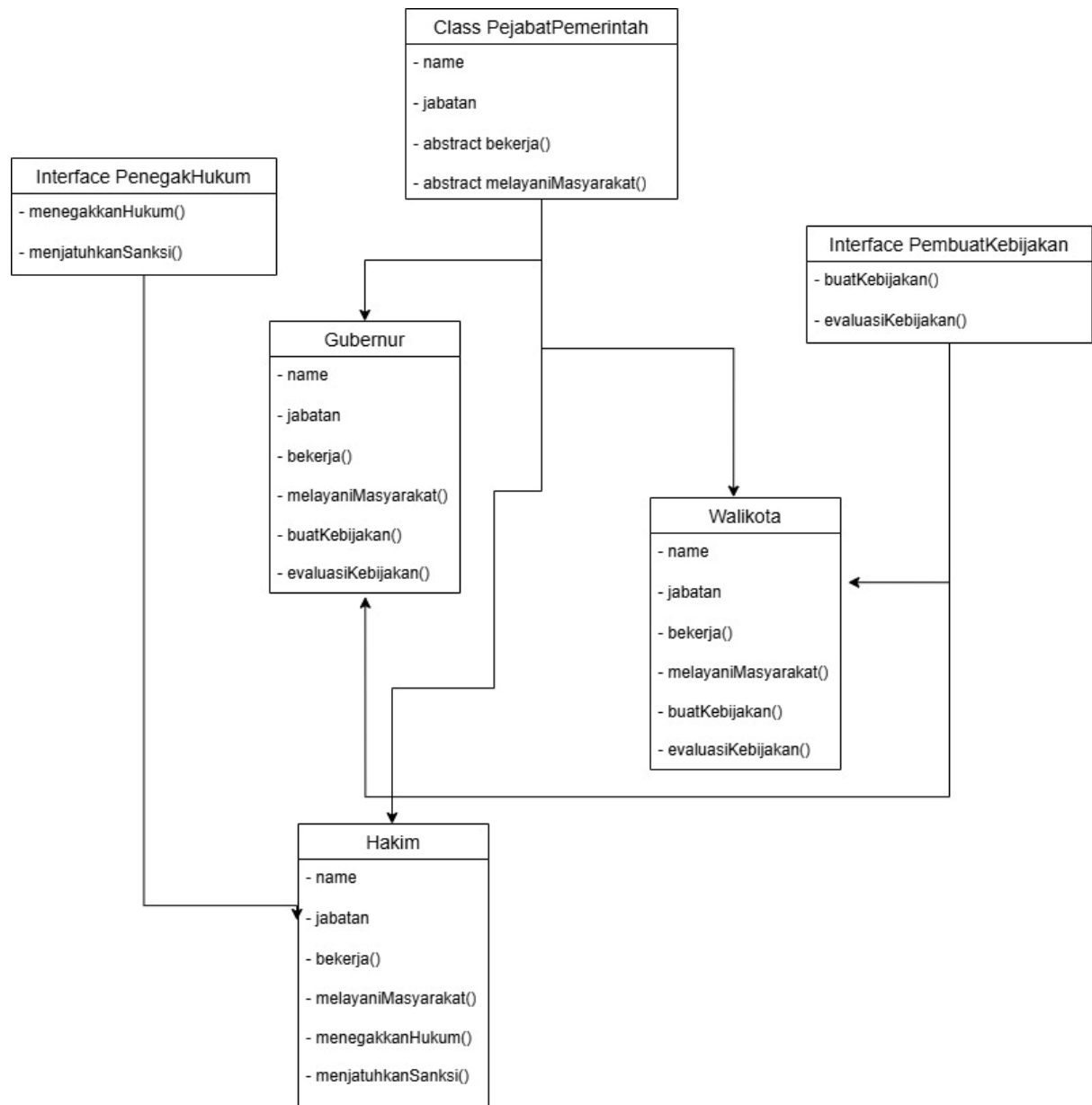
PROGRAM STUDI SARJANA TERAPAN
TEKNIK INFORMATIKA
POLITEKNIK NEGERI BANDUNG

2025

DAFTAR ISI

DAFTAR ISI.....	2
https://github.com/mailvlous/teknikPemrograman/tree/main/Week2/Tugas... Error! Bookmark not defined.	
1. Inheritance.....	3
2. Abstract Class	6
3. Interface	6
4. Polymorphism	7
Link Github.....	8

<https://github.com/mailvlous/teknikPemrograman/tree/main/Week4/PejabatPemerintah>



1. Inheritance

Konsep Inheritance disini terletak pada class PejabatPemerintah yang mewarisi class Gubernur, Hakim, dan Presiden.

```
PejabatPemerintah > J PejabatPemerintah.java > Language Support for Java(TM) by Red Hat > 🚧 PejabatPemerintah
Codeium: Refactor | Explain
1  abstract class PejabatPemerintah {
2      String nama;
3      String jabatan;
4
5      PejabatPemerintah(String nama, String jabatan) {
6          this.nama = nama;
7          this.jabatan = jabatan;
8      }
9
10     abstract void bekerja();
11     abstract void melayaniMasyarakat();
12 }
```

Di Class Presiden:

```
Codeium: Refactor | Explain
2  public class Presiden extends PejabatPemerintah implements PembuatKebijakan {
3      public Presiden(String nama, String jabatan) {
4          super(nama, jabatan);
5      }
6
7      Codeium: Refactor | Explain | Generate Javadoc | ✕
8      @Override
9      void bekerja() {
10         System.out.println(nama + " memimpin negara dan menghadiri pertemuan internasional.");
11     }
12
13     Codeium: Refactor | Explain | Generate Javadoc | ✕
14     @Override
15     void melayaniMasyarakat() {
16         System.out.println(nama + " memberikan pidato kepada rakyat.");
17     }
18
19     Codeium: Refactor | Explain | Generate Javadoc | ✕
20     @Override
21     public void buatKebijakan() {
22         System.out.println(nama + " membuat kebijakan baru.");
23     }
24
25     Codeium: Refactor | Explain | Generate Javadoc | ✕
26     @Override
27     public void evaluasiKebijakan() {
28         System.out.println(nama + " melakukan evaluasi kebijakan.");
29     }
30 }
```

Presiden diwarisi oleh Class PejabatPemerintah sehingga memiliki attribut:

- Nama
- Jabatan

Juga konsep abstraction pada fungsi:

- melayaniMasyarakat()
- bekerja()

Di Class Gubernur:

```
PejabatPemerintah > J Gubernur.java > Language Support for Java(TM) by Red Hat > Gubernur
Codeium: Refactor | Explain
1 public class Gubernur extends PejabatPemerintah implements PembuatKebijakan {
2     public Gubernur(String nama, String jabatan) {
3         super(nama, jabatan);
4     }
5
6     Codeium: Refactor | Explain | Generate Javadoc | X
7     @Override
8     void melayaniMasyarakat() {
9         System.out.println(x:"Gubernur sedang melayani masyarakat.");
10    }
11
12    Codeium: Refactor | Explain | Generate Javadoc | X
13    @Override
14    void bekerja() {
15        System.out.println(x:"Gubernur sedang bekerja.");
16    }
17
18    Codeium: Refactor | Explain | Generate Javadoc | X
19    @Override
20    public void buatKebijakan() {
21        System.out.println(x:"Gubernur membuat kebijakan baru.");
22    }
23
24    Codeium: Refactor | Explain | Generate Javadoc | X
25    @Override
26    public void evaluasiKebijakan() {
27        System.out.println(x:"Gubernur melakukan evaluasi kebijakan.");
28    }
29 }
```

Gubernur diwarisi oleh Class PejabatPemerintah sehingga memiliki attribut:

- Nama
- Jabatan

Juga konsep abstraction pada fungsi:

- melayaniMasyarakat()
- bekerja()

Di Class Hakim:

```
WEEK4
└─ .dist
   └─ PejabatPemerintah
       └─ Class
           └─ Gubernur.java
           └─ Hakim.java
           └─ PejabatPemerintah.java
           └─ PembuatKebijakan.java
           └─ PenegakHukum.java
           └─ Presiden.java
       └─ TekProWeek4.drawio

PejabatPemerintah > J Hakim.java > Language Support for Java(TM) by Red Hat > Hakim > menjatuhkanSanksi()
Codeium: Refactor | Explain
1 public class Hakim extends PejabatPemerintah implements PenegakHukum {
2     public Hakim(String nama, String jabatan) {
3         super(nama, jabatan);
4     }
5
6     Codeium: Refactor | Explain | Generate Javadoc | X
7     @Override
8     void bekerja() {
9         System.out.println(nama + " sedang bekerja sebagai hakim.");
10    }
11
12    Codeium: Refactor | Explain | Generate Javadoc | X
13    @Override
14    void melayaniMasyarakat() {
15        System.out.println(nama + " sedang melayani masyarakat.");
16    }
17
18    Codeium: Refactor | Explain | Generate Javadoc | X
19    @Override
20    public void menegakkanHukum() {
21        System.out.println(nama + " menegakkan hukum.");
22    }
23
24    Codeium: Refactor | Explain | Generate Javadoc | X
25    @Override
26    public void menjatuhkanSanksi() {
27        System.out.println(nama + " menjatuhkan sanksi.");
28    }
29 }
```

Hakim diwarisi oleh Class PejabatPemerintah sehingga memiliki attribut:

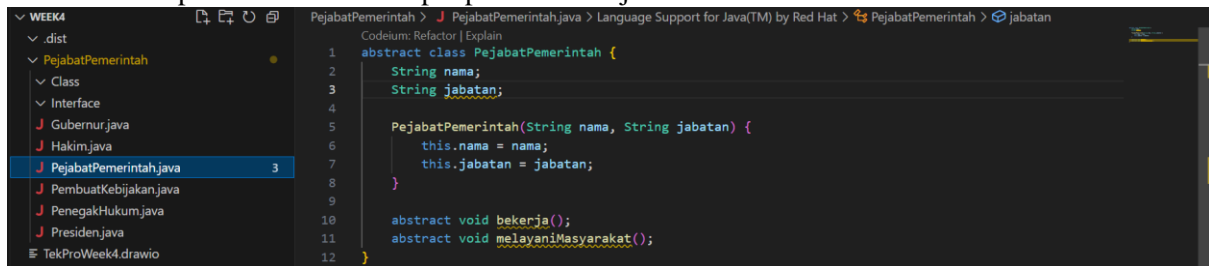
- Nama
- Jabatan

Juga konsep abstraction pada fungsi:

- melayaniMasyarakat()
- bekerja()

2. Abstract Class

Konsep abstract class terdapat pada class PejabatPemerintah itu sendiri

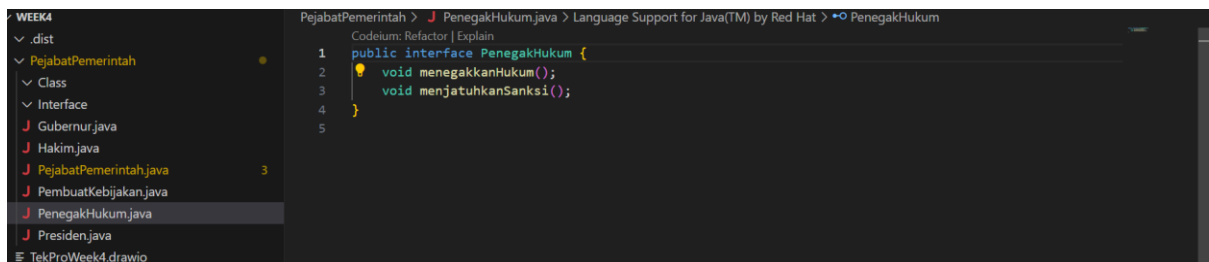


```
1 abstract class PejabatPemerintah {
2     String nama;
3     String jabatan;
4
5     PejabatPemerintah(String nama, String jabatan) {
6         this.nama = nama;
7         this.jabatan = jabatan;
8     }
9
10    abstract void bekerja();
11    abstract void melayaniMasyarakat();
12 }
```

3. Interface

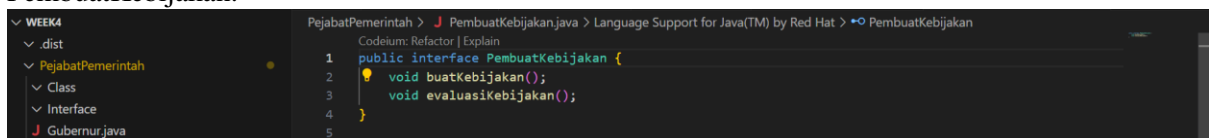
Konsep interface disini memiliki 2 buah interface

PenegakHukum:



```
1 public interface PenegakHukum {
2     void menegakkanHukum();
3     void menjatuhkanSanksi();
4 }
5
```

PembuatKebijakan:



```
1 public interface PembuatKebijakan {
2     void buatKebijakan();
3     void evaluasiKebijakan();
4 }
5
```

4. Polymorphism

Konsep polymorphism disini terdapat pada function

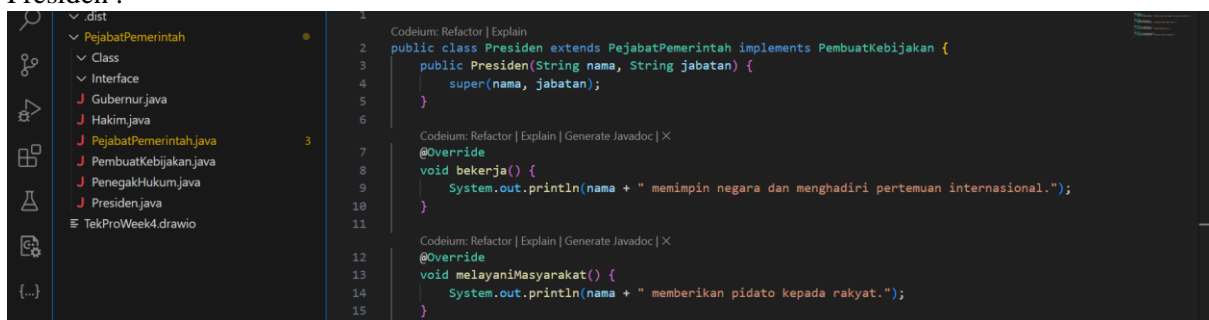
- bekerja()
- melayaniMasyarakat()



```
1 abstract class PejabatPemerintah {
2     String nama;
3     String jabatan;
4
5     PejabatPemerintah(String nama, String jabatan) {
6         this.nama = nama;
7         this.jabatan = jabatan;
8     }
9
10    abstract void bekerja();
11    abstract void melayaniMasyarakat();
12 }
```

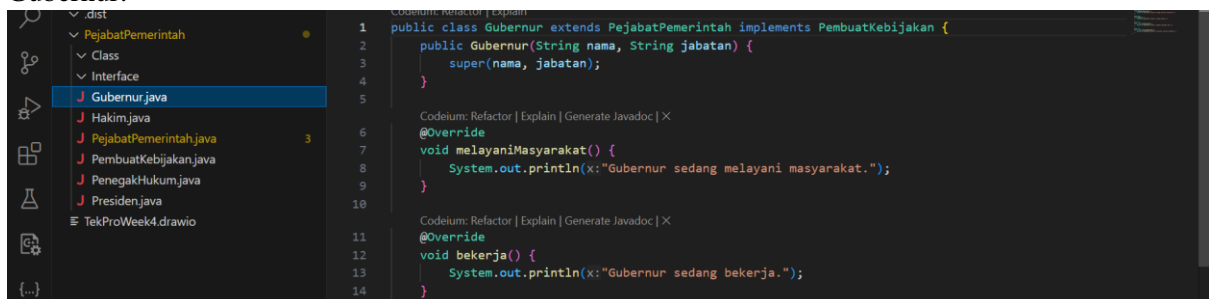
Yang dimana pada Presiden, Gubernur, dan Hakim mempunyai function bekerja() dan melayaniMasyarakat() yang berbeda beda

Presiden :



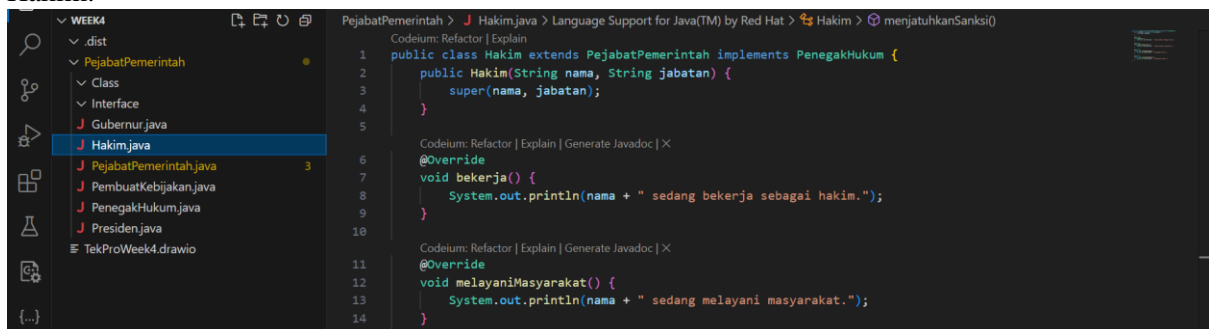
```
1 public class Presiden extends PejabatPemerintah implements PembuatKebijakan {
2     public Presiden(String nama, String jabatan) {
3         super(nama, jabatan);
4     }
5
6
7     @Override
8     void bekerja() {
9         System.out.println(nama + " memimpin negara dan menghadiri pertemuan internasional.");
10    }
11
12    @Override
13    void melayaniMasyarakat() {
14        System.out.println(nama + " memberikan pidato kepada rakyat.");
15    }
16 }
```

Gubernur:



```
1 public class Gubernur extends PejabatPemerintah implements PembuatKebijakan {
2     public Gubernur(String nama, String jabatan) {
3         super(nama, jabatan);
4     }
5
6
7     @Override
8     void melayaniMasyarakat() {
9         System.out.println(x:"Gubernur sedang melayani masyarakat.");
10    }
11
12    @Override
13    void bekerja() {
14        System.out.println(x:"Gubernur sedang bekerja.");
15    }
16 }
```

Hakim:



```
1 public class Hakim extends PejabatPemerintah implements PenegakHukum {
2     public Hakim(String nama, String jabatan) {
3         super(nama, jabatan);
4     }
5
6
7     @Override
8     void bekerja() {
9         System.out.println(nama + " sedang bekerja sebagai hakim.");
10    }
11
12    @Override
13    void melayaniMasyarakat() {
14        System.out.println(nama + " sedang melayani masyarakat.");
15    }
16 }
```

Link Github

<https://github.com/mailvlous/teknikPemrograman/tree/main/Week4/PejabatPemerintah>