**PRAKTIKUM-5 PDPL**

***STRUCTURAL PATTERN - PROXY***

**Mata Kuliah : Pola Desain Perangkat Lunak**

**Semester : 6 (Enam)**

**Dosen : Tifanny Nabarian, S.Kom., M.T.I.**

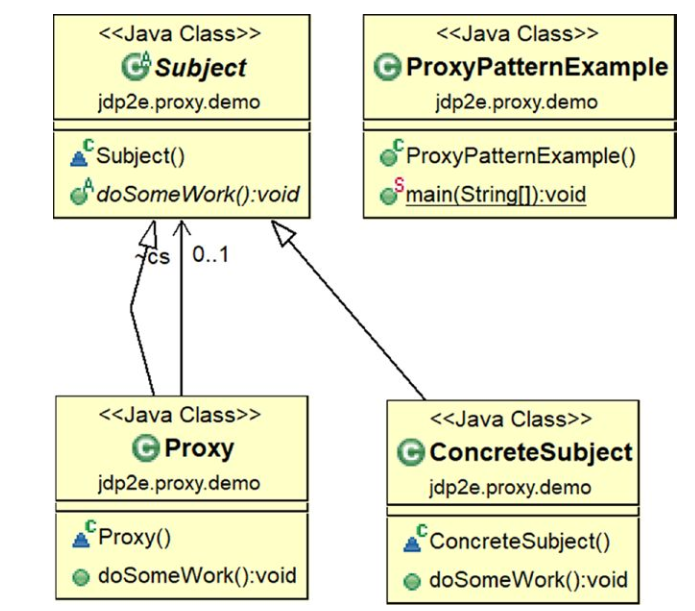
**Nama Mahasiswa : chairil hilman syah**

**NIM : 0110217054**

**Instruksi**

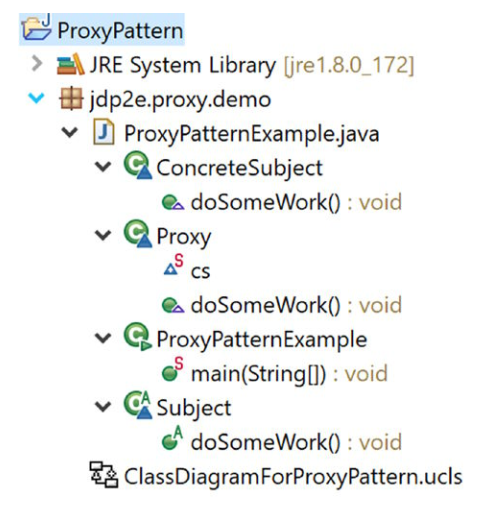
* Kerjakan tugas di bawah ini secara **individu.**
* Kerjakan secara **berurutan.**
* Modifikasi *source code*, dengan cara menambahkan **NIM** Anda pada setiap *caption* yang ditampilkan di *result* yang di*screenshot*.

1. **Perhatikan ilustrasi berikut ini:**



Gambar . Ilustrasi Relasi

1. **Pahamilah struktur package yang akan dibuat.**



Gambar . Package Explorer

1. **Buatlah Kelas Subject dan ConcreteSubject dengan source sebagai berikut:**

**Subject.java**

|  |
| --- |
| **package** jdp2e.proxy.demo;  **public** **abstract** **class** Subject {  **public** **abstract** **void** doSomeWork();  } |

**ConcreteSubject.java**

|  |
| --- |
| **package** jdp2e.proxy.demo;  **public** **class** ConcreteSubject **extends** Subject{  @Override  **public** **void** doSomeWork()  {  System.***out***.println("doSomeWork() inside ConcreteSubject is invoked.");  }  } |

1. **Lanjutkan dengan membuat kelas Proxy dengan sourcode sbb.,**

**Proxy.java**

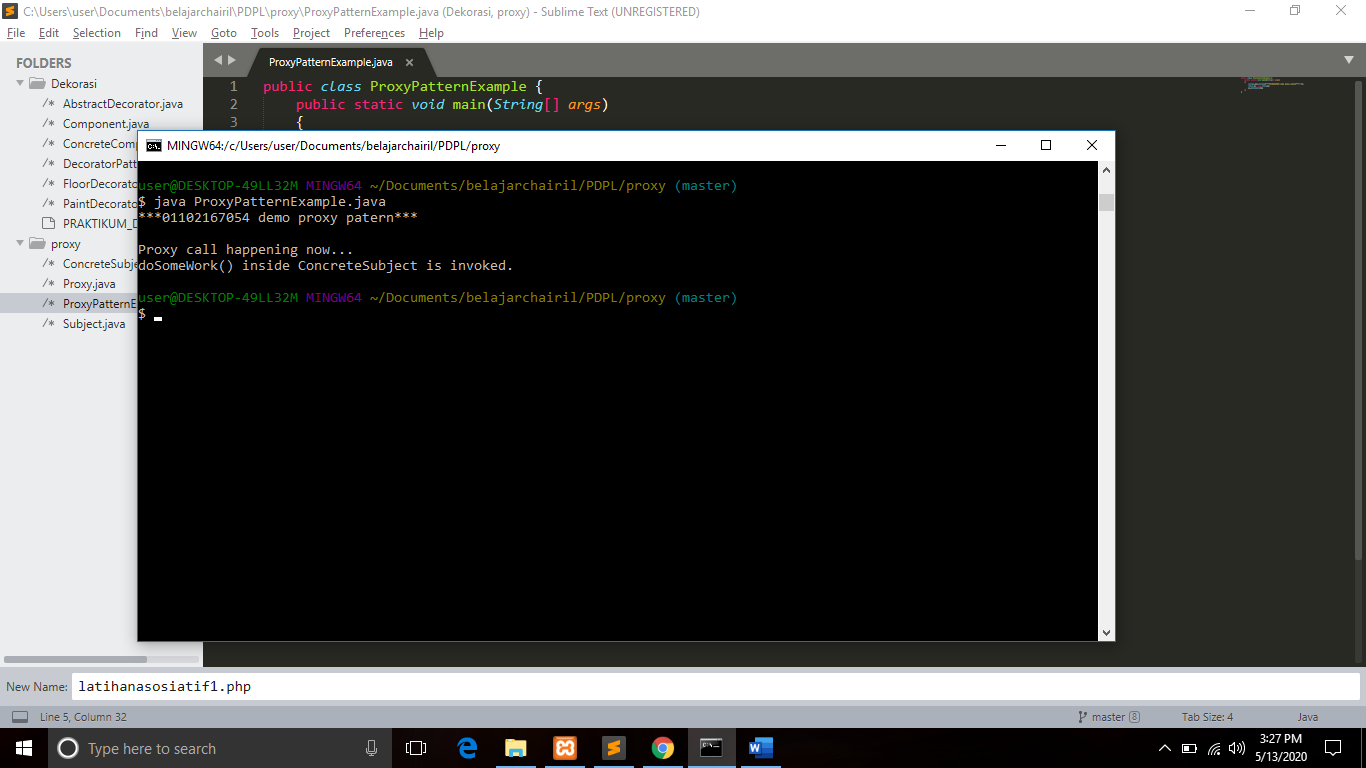
|  |
| --- |
| **package** jdp2e.proxy.demo;  **public** **class** Proxy **extends** Subject{  **static** Subject *cs*;  @Override  **public** **void** doSomeWork()  {  System.***out***.println("Proxy call happening now...");  //Lazy initialization:We'll not instantiate until the method is  //called  **if** (*cs* == **null**)  {  *cs* = **new** ConcreteSubject();  }  *cs*.doSomeWork();  }  } |

1. **Lanjutkan dengan membuat kelas client -> ProxyPatternExample, dengan source sbb.:**

**ProxyPatternExample.java**

|  |
| --- |
| **package** jdp2e.proxy.demo;  **public** **class** ProxyPatternExample {  **public** **static** **void** main(String[] args)  {  System.***out***.println("\*\*\*Proxy Pattern Demo\*\*\*\n");  Proxy px = **new** Proxy();  px.doSomeWork();  }  } |

1. **Update source pada langkah 5, dengan menampilkan NIM Anda. Kemudian jalankan *main* program tersebut.**



1. Berikan penjelasan terkait proses implementasi *Proxy* pada source 3,4 dan 5.

|  |
| --- |
| **Penjelasan pertahapan:**   1. **Tahap 3,** pada tahap ini kita membuat sebuah kelas dimana kelas tersebut adalah kelas subject yang ingin di tuju oleh pihak ketiga. Dan di tahap ini kita membuat pihak ketiganya yaitu concrete subjek. 2. **Tahap 4,** pada tahap ini kita membuat sebuah class proxy yang dmna nanti berfungsi seolah olah menjadi si pihak ketiga dan dia pun akan mengakses lagsung kepada subject dan nnti si client hanya tinggal mengakses si proxy ini tanpa harus akses ke si pihak ke tiga. 3. **Tahap 5,** pada tahap ini kita membuat class client yang nnti akan mengakses pihak ketiga namun nyatanya dia mengakes class proxy. |

1. Menurut pendapat Anda setelah mengimplementasikan *sourcode* di atas, apa perbedaan dari metode Structural Pattern **Adapter** dan Structural Pattern **Proxy**?

|  |
| --- |
| **dalam tujuan pola adaptor** adalah untuk mengubah antarmuka kelas / A ke harapan klien B tetapi tujuan dari pola **proxy** adalah ketika client B mengakses ke client A , client B nyatnya mengakses proxy yang ada hal tersebut digunakan karena beberapa alasan salah satunya hal tersebut mgkin terjadi karena client A memiliki ukurang yang sangat besar. |

**\*\*\***