**PRAKTIKUM-5 PDPL**

***STRUCTURAL PATTERN - PROXY***

**Mata Kuliah : Pola Desain Perangkat Lunak**

**Semester : 6 (Enam)**

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

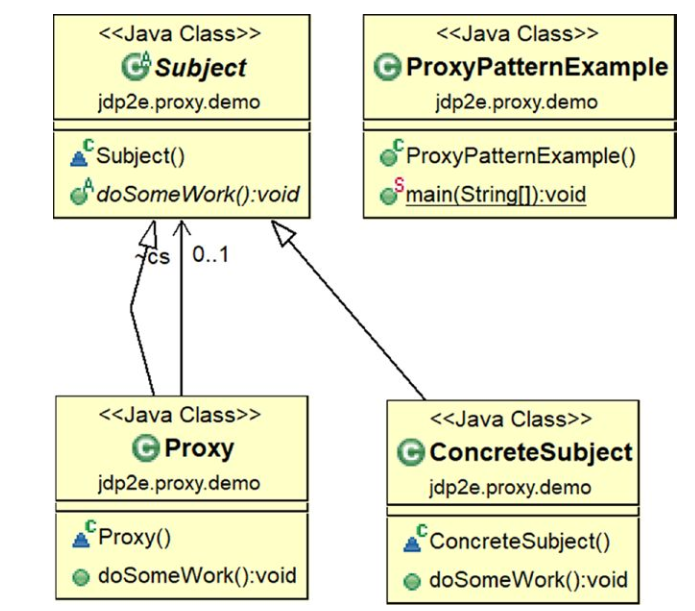
**Nama Mahasiswa : Muhammad Azhar Rasyad**

**NIM : 0110217029**

**Instruksi**

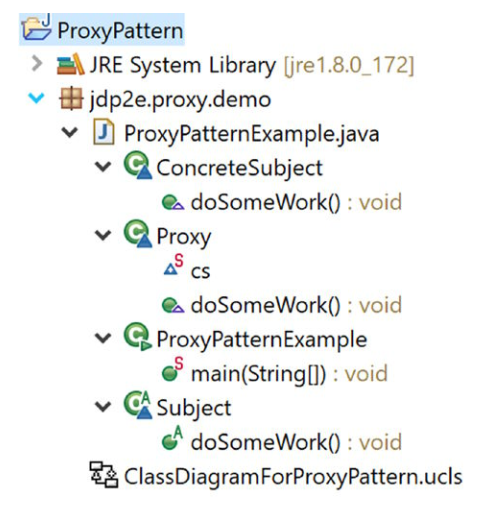
* Kerjakan tugas di bawah ini secara **individu.**
* Kerjakan secara **berurutan.**
* Modifikasi *source code*, dengan cara menambahkan **NIM** Anda pada setiap *caption* yang ditampilkan di *result* yang di*screenshot*.

1. **Perhatikan ilustrasi berikut ini:**



Gambar 1. Ilustrasi Relasi

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



Gambar 2. 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.**

|  |
| --- |
| *Screenshot Hasil Running Main Program* |

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

|  |
| --- |
| Awalnya ada sebuah client yang mengakses kelas Proxy dengan **method doSomeWork()**, seperti kode dibawah :    Method doSomeWork() tersebut diakses melalui kelas Proxy dengan kode berikut :    Pada kode di atas terlihat bahwa method doSomeWork() memiliki beberapa perintah, yang pertama yaitu menginformasikan **"Proxy call happening now..."**. Informasi tersebut untuk mengabarkan kalau panggilan proxy sedang terjadi sekarang, panggilan yang dimaksud ditujukan untuk memanggil kelas Subject dan mengerjakan suatu pekerjaan seperti berikut :    Akan tetapi jika kelas Subject tersebut sedang tidak ada maka digantikan oleh kelas lain, dalam kasus ini yang digantikan oleh ConcreteSubject dengan kode sebelumnya pada method doSomeWork() :    Jika dilihat kode program di atas pada “if (cs == null)” yang dimana cs itu adalah Subject, jika cs itu null maka digantikan oleh kelas ConcreteSubject yang berisi kode program berikut :    dan akan menampilkan informasi berikut "doSomeWork() inside ConcreteSubject is invoked.”, sehingga proses tersebut dinamakan Proxy karena kelas yang sebelumnya ingin dipanggil tidak ada maka digantikan oleh kelas yang lainnya  Kembali ke kode pada client, maka pada px.doSomeWork() yang diprint adalah kelas ConcreteSubject dan bukan kelas Subject    Berikut hasilnya |

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

|  |
| --- |
| Menurut saya, Adapter dan Proxy memang memiliki kesamaan dalam hal menggantikan suatu task, akan tetapi perbedaan utama dari Adapter dan Proxy yaitu jikalau Adapter menggantikan suatu task, maka task yang digantikan tersebut memang tidak bisa menjalankan task tersebut akan tetapi task tersebut masih berjalan, sedangkan Proxy menggantikan suatu task ketika suatu task tersebut sedang tidak ada dan bukan tidak bisa menjalankan task tersebut. |

**\*\*\***