**PRAKTIKUM-7 PDPL**

***BEHAVIORAL PATTERN – CHAIN OF RESPONSIBILITY***

**Mata Kuliah : Pola Desain Perangkat Lunak**

**Semester : 6 (Enam)**

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

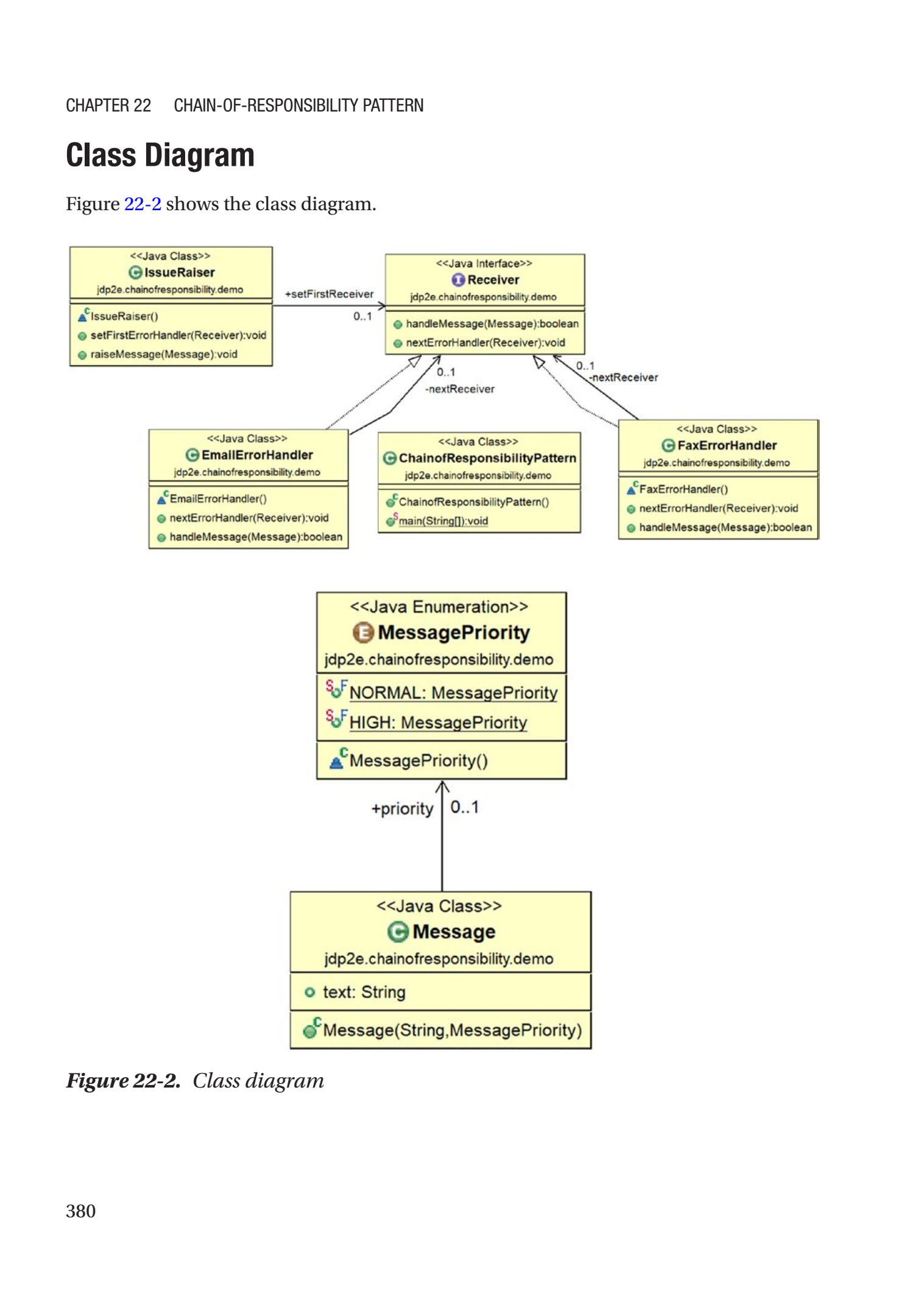
**Nama Mahasiswa : Muhammad Azhar Rasyad**

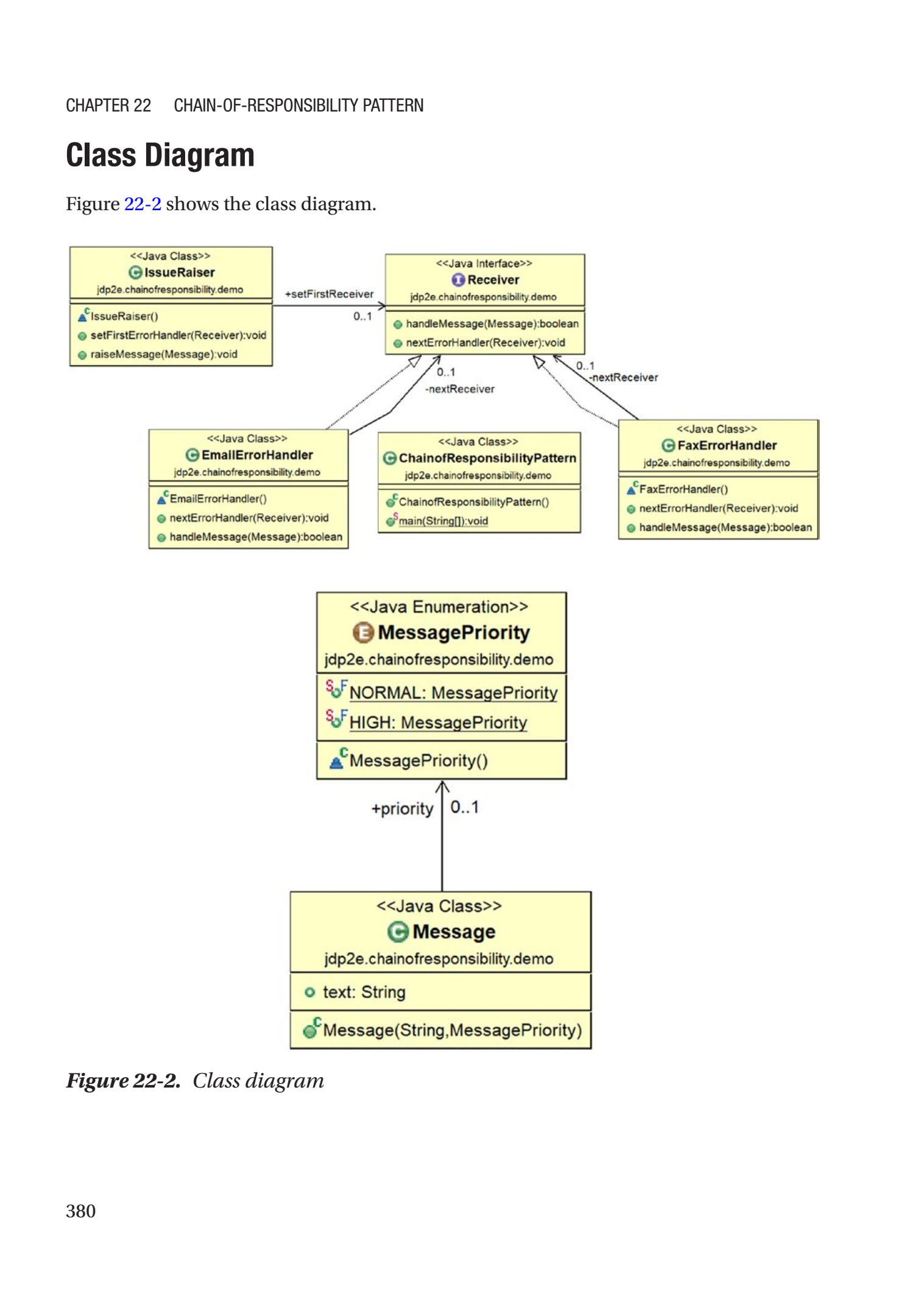
**NIM : 0110217029**

**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*.

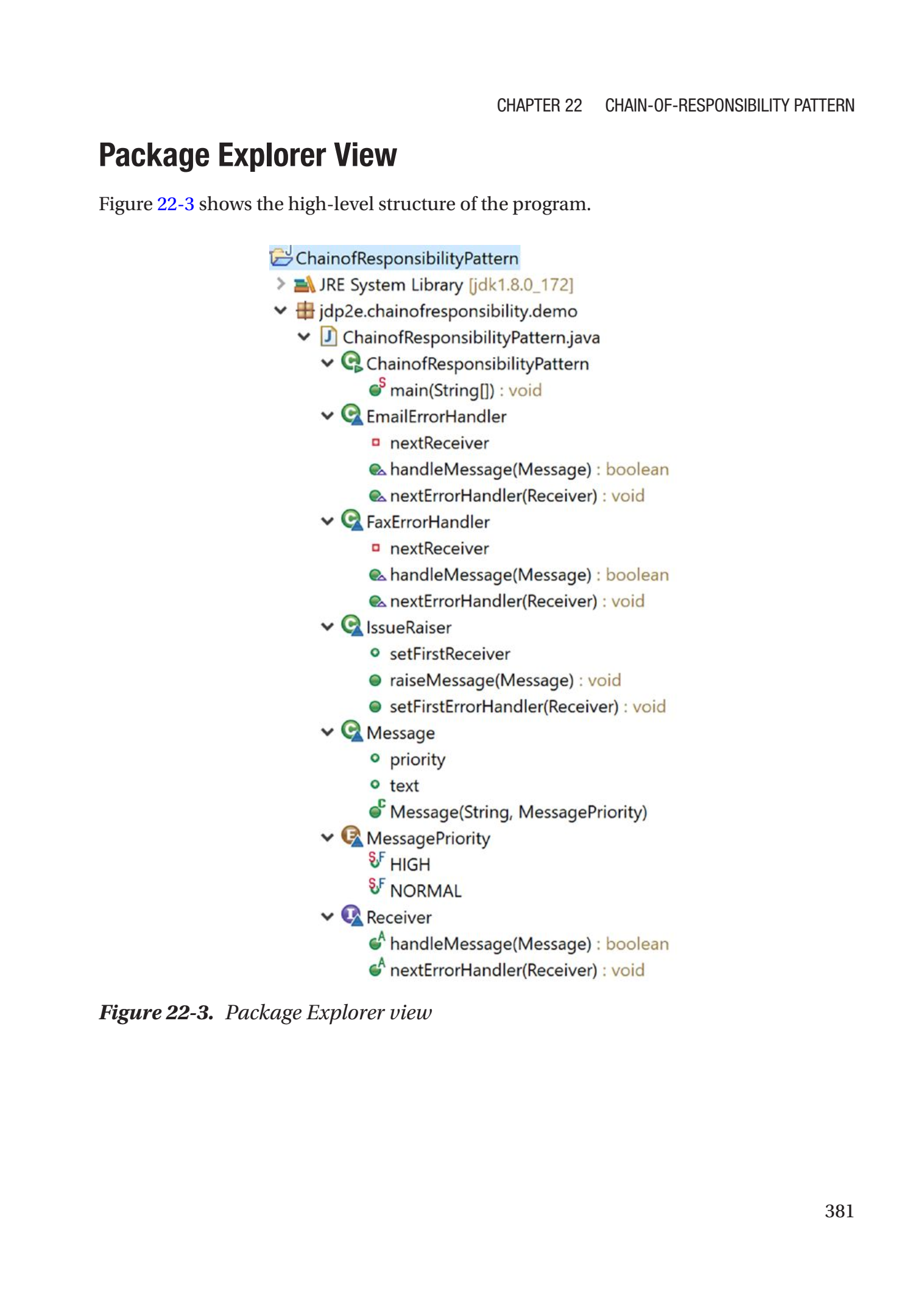
1. **Perhatikan ilustrasi berikut ini:**





Gambar 1. Ilustrasi Relasi

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



Gambar 2. Package Explorer

1. **Buatlah Kelas MessagePriority dan Message dengan source sebagai berikut:**

**MessagePriority.java**

|  |
| --- |
| **package** jdp2e.chainofresponsibility.demo;  **public** **enum** MessagePriority {  ***NORMAL***,  ***HIGH***  } |

**Message.java**

|  |
| --- |
| **package** jdp2e.chainofresponsibility.demo;  **public** **class** Message {  **public** String text;  **public** MessagePriority priority;  **public** Message(String msg, MessagePriority p)  {  text = msg;  **this**.priority = p;  }  } |

1. **Lanjutkan dengan membuat interface Receiver, kelas IssueRaiser, FaxErrorHandler dan EmailErrorHandler dengan sourcode sbb.,**

**Receiver.java**

|  |
| --- |
| **package** jdp2e.chainofresponsibility.demo;  **public** **interface** Receiver {  **boolean** handleMessage(Message message);  **void** nextErrorHandler(Receiver nextReceiver);  } |

**IssueRaiser.java**

|  |
| --- |
| **package** jdp2e.chainofresponsibility.demo;  **public** **class** IssueRaiser {    **public** Receiver setFirstReceiver;    **public** **void** setFirstErrorHandler(Receiver firstErrorHandler)  {  **this**.setFirstReceiver = firstErrorHandler;  }  **public** **void** raiseMessage(Message message)  {  **if** (setFirstReceiver != **null**)  setFirstReceiver.handleMessage(message);  }  } |

**FaxErrorHandler.java**

|  |
| --- |
| **package** jdp2e.chainofresponsibility.demo;  **public** **class** FaxErrorHandler **implements** Receiver{  **private** Receiver nextReceiver;  @Override  **public** **void** nextErrorHandler(Receiver nextReceiver)  {  **this**.nextReceiver = nextReceiver;  }    @Override  **public** **boolean** handleMessage(Message message)  {  **if** (message.text.contains("Fax"))  {    System.***out***.println(" FaxErrorHandler processed " +message.priority +" priority issue :"+ message.text);  **return** **true**;  }  **else**  {  **if** (nextReceiver != **null**)  nextReceiver.handleMessage(message);  }  **return** **false**;  }  } |

**EmailErrorHandler.java**

|  |
| --- |
| **package** jdp2e.chainofresponsibility.demo;  **public** **class** EmailErrorHandler **implements** Receiver{  **private** Receiver nextReceiver;    @Override  **public** **void** nextErrorHandler(Receiver nextReceiver)  {  **this**.nextReceiver = nextReceiver;  }    @Override  **public** **boolean** handleMessage(Message message)  {  **if** (message.text.contains("Email"))  {  System.***out***.println(" EmailErrorHandler processed "+message.  priority+ " priority issue: "+message.text);  **return** **true**;  }  **else**  {  **if** (nextReceiver != **null**)  nextReceiver.handleMessage(message);  }  **return** **false**;  }  } |

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

**ChainofResponsibilityPattern.java**

|  |
| --- |
| **package** jdp2e.chainofresponsibility.demo;  **public** **class** ChainofResponsibilityPattern {  **public** **static** **void** main(String[] args)  {    System.***out***.println("\n \*\*\*Chain of Responsibility Pattern Demo\*\*\*\n");    /\* Forming the chain as IssueRaiser->FaxErrorhandler->  EmailErrorHandler\*/  Receiver faxHandler, emailHandler;    //Objects of the chains  IssueRaiser issueRaiser = **new** IssueRaiser();  faxHandler = **new** FaxErrorHandler();  emailHandler = **new** EmailErrorHandler();    //Making the chain  //Starting point:IssueRaiser will raise issues and set the first  //handler  issueRaiser.setFirstErrorHandler(faxHandler);    //FaxErrorHandler will pass the error to EmailHandler if needed.  faxHandler.nextErrorHandler(emailHandler);    //EmailErrorHandler will be placed at the last position in the chain  emailHandler.nextErrorHandler(**null**);    Message m1 = **new** Message("Fax is going slow.",MessagePriority.***NORMAL***);  Message m2 = **new** Message("Emails are not reaching.", MessagePriority.***HIGH***);  Message m3 = **new** Message("In Email, CC field is disabled always.", MessagePriority.***NORMAL***);  Message m4 = **new** Message("Fax is not reaching destinations.", MessagePriority.***HIGH***);  issueRaiser.raiseMessage(m1);  issueRaiser.raiseMessage(m2);  issueRaiser.raiseMessage(m3);  issueRaiser.raiseMessage(m4);  }  } |

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 *Chain of Responsibility* pada source 3,4 dan 5.

|  |
| --- |
| Dalam implementasi chain of responbility langkah awalnya yaitu membuat rantai atau chain dari ujung awal ke ujung akhir. Pada kelas ChainofResponsibilityPattern dibuatlah rantai tersebut diantaranya:   * IssueRaiser sebagai ujung awal sebuah rantai jika ada sebuah request dan meneruskan request tersebut untuk ditangani pertama kali oleh FaxErrorhandler * FaxErrorhandler sebagai rantai kedua setelah IssueRaiser untuk menangani request yang tidak bisa ditangani ranti sebelumnya akan tetapi jika masih tidak dapat ditangani maka akan diteruskan ke rantai selanjutnya * EmailErrorHandler sebagai ujung akhir sebuah rantai yang apabila tidak bisa menangani request dari rantai sebelumnya   Jika diilustrasikan maka bentuk rantai tersebut menjadi seperti berikut  **IssueRaiser -> FaxErrorhandler -> EmailErrorHandler**  Setelah dibuat rantai tersebut maka tinggal menunggu adanya request, dalam kasus ini request berupa kelas Message yang menyampaikan adanya error tentang Fax atau Email namun selain tentang itu maka request tidak dapat ditangani.  Kasus kali ini terdapat empat request yaitu:    Dapat dilihat pada kode program tersebut, terdapat kata kunci yaitu Fax atau Email dan ketika kedua kata kunci tersebut muncul maka akan segera ditangani oleh errorhandler masing-masing, request Fax oleh FaxErrorhandler dan request Email oleh EmailErrorHandler akan tetapi masih terdapat celah yaitu ketika terdapat request dengan kata kunci Fax dan Email bersamaan maka hanya ditangani oleh FaxErrorhandler, mengapa? Karena dalam rantai sebelumnya FaxErrorhandler lebih dahulu dibanding EmailErrorHandler yang dibagian ujung.  Ketika Message sudah ditangani oleh errorhandler yang tepat maka masing-masing errorhandler tersebut akan memberikan pemberitahuan bahwa Message sedang diproses dan diberikan MessagePriority disetiap requestnya dengan tujuan disolve terlebih dahulu priority HIGH dibanding priority NORMAL  Lalu bagaimana request tersebut dapat mengetahui errorhandler yang tepat? Sebelumnya telah dijelaskan kata kunci Fax dan Email, maka dalam kelas FaxErrorhandler terdapat kata kunci tersebut apabila dalam suatu request terdapat kata kunci Fax maka FaxErrorhandler akan menangani request tersebut akan tetapi tidak terdapat kata kunci Fax maka akan melanjutkan requestnya ke errorhandler selanjutnya yaitu EmailErrorHandler. Sama seperti FaxErrorhandler, dalam EmailErrorHandler terdapat kata kunci Email yang jika ada dalam request maka dapat diproses namun jika tidak ada juga maka request tersebut akan berhenti karena sudah berada diujung rantai ***(end of chain)***    Dapat dilihat pada kode di atas bahwa request pertama ditangani oleh IssueRaiser selanjutnya FaxErrorhandler dan terakhir EmailErrorHandler namun setelah itu tidak ada lagi penanganan errorhandler. |

1. Bagaimana cara object issueRaiser mengetahui bahwa *request* yang dikirim diproses oleh *errorhandler* yang terakhir (***end of chain***)?

|  |
| --- |
| Sebelumnya setiap errorhandler diinisiasi dengan kelas Receiver, tujuannya agar membuat ranti penghubung antara setiap errorhandler tersebut, seperti pada kode berikut    Sementara pada kelas IssueRaiser karena ditetapkan sebagai ujung awal sebuah rantai, maka dapat dipastikan bahwa request akan langsung masuk ke IssueRaiser dan otomatis diketahui oleh IssueRaiser melalui kode berikut    Lalu bagaimana IssueRaiser mengetahui bahwa requestnya telah mencapai ***end of chain*** atau tidak sementara ia berada pada ***start of chain***? Untuk mengetahuinya dengan method raiseMessage pada IssueRaiser, terlihat pada kode berikut    Kode diatas yang menunjukkan bahwa IssueRaiser dapat mengetahui request telah sampai di errorhandler yang mana, ketika request dapat ditangani oleh FaxErrorhandler maka akan mengembalikan nilai true kembali ke raiseMessage dan IssueRaiser mengetahui bahwa telah ditangani oleh FaxErrorhandler  Sama halnya ketika EmailErrorHandler telah berhasil menangani request yang ada, maka akan mengembalikan nilai true kembali ke raiseMessage di IssueRaiser dan jika request yang ditangani oleh kedua errorhandler tersebut tidak dapat ditangani maka akan mengembalikan nilai false kembali ke raiseMessage  Itulah mengapa IssueRaiser mengetahui sampai dimana request ditangani oleh errorhandler (Menurut saya) |

**\*\*\***