# ASSIGNMENT PROJECT


| | | |
|---|---|---|
| **Anggota Kelompok** | **:** | **(1) Fendy Wijaya (NIM: 2602092150)** |
| | | **(2) Tiara Intan Kusuma (NIM: 2602172220)** |
| **Kelas** | **:** | **LT01** |
| **Mata Kuliah** | **:** | **COMP6820001 – Object Oriented Programming** |
| **Dosen Pengampu** | **:** | **D6797 – Adi Wibowo P., S.Tr.Kom., M.Kom.** |


## Instructions:

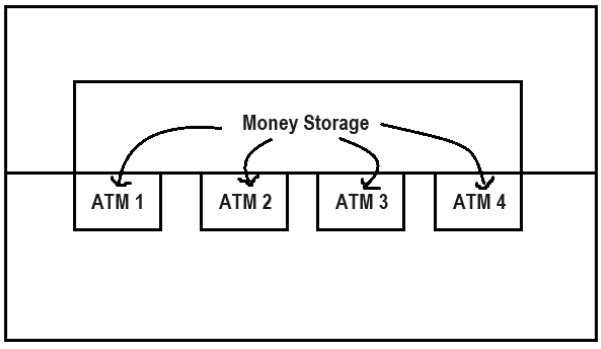
- Create Team (2- 3 members)
- Build Application including this feature :
  - Multithreading
  - Class Relation (Composition / Aggregation / Association)
- Min. 3 Menu
- Topic, select one from : **Book Library App, Showroom App, ECommerce App, ATM Simulator**.
- Submit on Forum :
  - Source Code in Zip
  - Project Description including Application Feature, Multithreading part, Relation between Class.
  - Screenshots of Console Application
- Deadline based on Forum.

## A. Project Descriptions

Pada penugasan proyek kali ini, kami memilih topik **ATM Simulator** untuk dibuatkan menjadi sebuah aplikasi console dengan bahasa pemrograman Java. Berikut ini kami sajikan skenario yang menjadi landasan awal pembuatan program simulator ATM ini.

Fendy adalah seorang pebisnis muda. Ia baru saja mendirikan bank baru bernama Bank Fentira. Baginya, kenyamanan dan keamanan nasabah merupakan urutan pertama prioritasnya dalam merintis usaha perbankannya. Fendy menyadari bahwa tingkat kriminalitas berupa pencurian dan pembobolan bank masih sering terjadi di Indonesia. Di tahun 2015 saja, data CNN menunjukkan bahwa sepertiga kasus penipuan ATM dunia terjadi di Indonesia. Ditambah lagi, modus kejahatan juga semakin berkembang seiring perkembangan zaman dan teknologi, sehingga kewaspadaan dan antisipasi merupakan tindakan preventif yang wajib diperhatikan oleh nasabah maupun penyedia layanan perbankan. Maka dari itu, Fendy ingin membuat sebuah program yang dapat memantau setiap perputaran uang di mesin ATM yang terdapat di banknya.

Misalkan Fendy telah membeli 4 buah mesin ATM yang akan diletakkan di bagian depan banknya. Keunikan yang dimiliki oleh bank ini adalah cara kerja mesin ATM yang tidak memiliki penyimpan uang sendiri di masing-masing mesinnya. Setiap mesin ATM didesain sedemikian rupa sehingga keempat mesin tersebut mengakses tempat penyimpanan uang yang sama, dan setiap mesin dapat bekerja bersamaan mengambil uang dari tempat penyimpanan uang tersebut. Tujuannya agar setiap transaksi dapat dicatat dengan mudah dari aplikasi simulator. Skema struktur ATM dan penyimpanan uang digambarkan sebagai berikut.



Program simulator yang akan dibuat harus menunjukkan secara detail setiap transaksi yang terjadi pada keempat mesin ATM tersebut, mulai dari kondisi mesin, jumlah uang di dalam penyimpanan, dan riwayat transaksi yang terjadi di setiap mesin ATM. Administrator aplikasi harus bisa melakukan perubahan-perubahan seperti memasukkan sejumlah uang ke dalam penyimpanan, ataupun menghapus riwayat transaksi yang tercatat jika diperlukan.

## B. Application Features

Untuk aplikasi simulator ATM ini, kami akan membangun aplikasi dengan fitur-fitur yang kami implementasikan dalam bentuk menu programming. Berikut ini adalah menu yang akan ditampilkan dalam aplikasi beserta penjelasan rincinya.

*(1) Masukkan uang ke storage ATM*

Pada fitur ini, simulator akan menghitung secara otomatis jumlah uang yang terdapat pada ruang penyimpanan (storage) ATM. Lalu, fitur ini akan menambahkan sejumlah uang ke dalam storage sesuai perintah administrator. Untuk alasan konsistensi, mesin hanya akan menerima uang pecahan Rp100.000,- saja untuk setiap transaksi, sehingga tidak boleh ada uang pecahan lain selain Rp100.000,-, termasuk ketika administrator memasukkan uang ke dalam storage.

*(2) Pantau penarikan setiap mesin ATM*

Fitur ini men-trigger ATM untuk melakukan pemantauan setiap transaksi yang beriringan terjadi di setiap mesin ATM. Setiap transaksi akan di-print di console, mulai saat mesin sedang memproses sampai mesin selesai mengeluarkan uang. Setiap mesin berjalan bersamaan di waktu yang sama, sehingga jika uang habis di dalam storage, semua mesin yang ingin mengakses storage akan mengeluarkan notifikasi yang sama terkait habisnya uang di dalam storage. Fitur ini tidak dapat berhenti sampai seluruh uang di dalam storage habis.

*(3) Kosongkan uang dari storage ATM*

Fitur ini diperuntukkan hanya bagi administrator ketika ia ingin menarik semua uang yang telah ia masukkan ke dalam storage. Fitur ini dipakai jika sesuatu

hal yang fatal terjadi, misalnya perampokan atau kesalahan perhitungan uang. Dengan menjalankan fitur ini, uang yang sudah dimasukkan administrator ke dalam storage akan ditarik kembali sehingga nilai uang di dalam storage menjadi Rp0,-.

*(4) Tampilkan log transaksi setiap mesin ATM*

Pada fitur ini, setiap riwayat transaksi dari keempat mesin ATM ketika dijalankan pada fitur kedua akan dicatat dan ditampilkan oleh aplikasi. Data pada fitur ini akan membantu melacak setiap transaksi yang terjadi, sehingga dapat diketahui frekuensi transaksi setiap mesin dan total uang yang dikeluarkan oleh mesin tersebut. Jika frekuensi transaksi mesin tidak merata secara drastis, kemungkinan ada masalah pada mesin tersebut.

*(5) Hapus log transaksi semua mesin ATM*

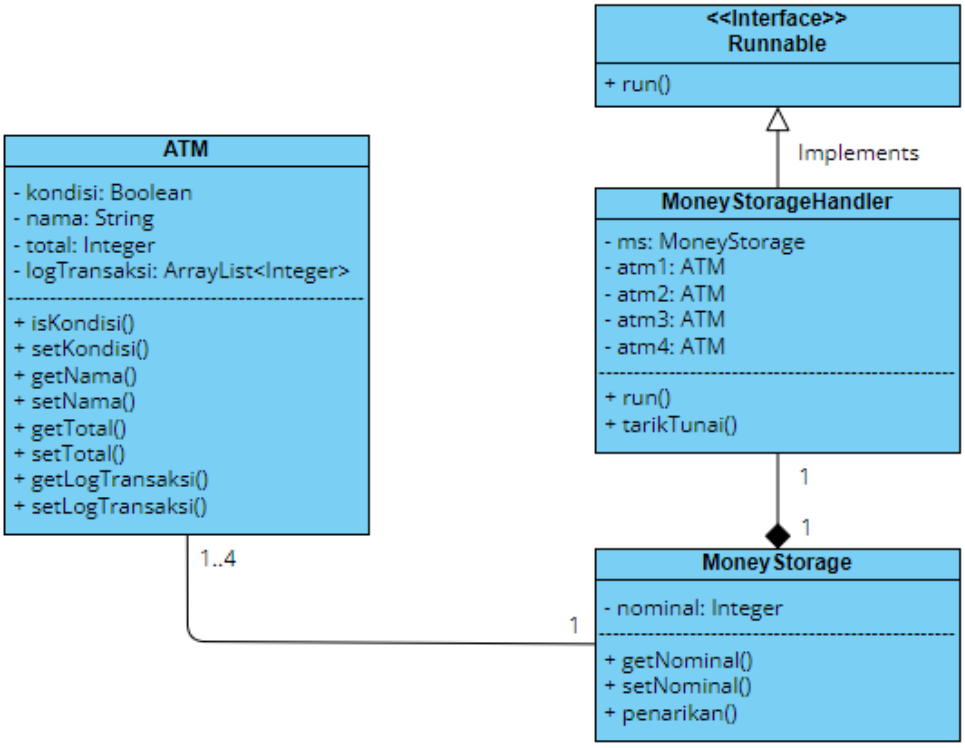
Untuk alasan efisiensi memori, terkadang log transaksi yang sudah di-backup dapat dihapus agar aplikasi tidak semakin berat. Oleh karena itu, administrator dapat menghapus log transaksi jika sudah tidak diperlukan. Akan tetapi, untuk mencegah penghapusan data secara tidak disengaja, perlu ada validasi untuk meyakinkan bahwa data log transaksi benar-benar akan dihapus. Data log transaksi keempat ATM akan dihapus dan dikembalikan ke nilai nol.

*(6) Keluar dari program simulasi*

Terakhir, untuk keluar dari looping program simulasi yang sedang berjalan, administrator dapat menekan tombol nomor 6 untuk menghentikan aplikasi simulator ATM ini.

## C. Class Relationships

Terdapat empat kelas yang dibuat dalam aplikasi ini, yaitu kelas Main sebagai kelas utama tempat program dijalankan, kelas ATM, kelas MoneyStorage, dan kelas MoneyStorageHandler. Berikut ini disajikan hubungan antarkelas (selain Main class) dalam bentuk class diagram. Class diagram berikut juga akan menyertakan semua atribut dan method yang difungsikan di dalam setiap kelas yang dibuat.




Pada class diagram tersebut, terlihat bahwa kelas ATM memiliki hubungan **association** dengan kelas MoneyStorage. Hubungan keduanya termasuk association biasa karena objek ATM di dunia nyata tidak memiliki hubungan kepemilikan khusus dengan objek tempat penyimpanan uang (storage). Keduanya berdiri sendiri-sendiri, meskipun saling terhubung secara fisik. ATM tidak memiliki penyimpan uang sendiri, dan penyimpan uang juga tidak memiliki mesin sendiri, sehingga relasi yang terbentuk di antara kedua kelas ini hanya bersifat asosiasi semata. Satu mesin ATM bisa mengakses satu penyimpan uang bersama dengan ATM lainnya, dan satu penyimpan uang bisa diakses oleh 1-4 mesin ATM yang tersedia di Bank Fentira.

Selain itu, kelas MoneyStorage memiliki hubungan **composition** dengan kelas MoneyStorageHandler. Hubungan keduanya bersifat kepemilikan, karena MoneyStorageHandler harus dimiliki oleh MoneyStorage agar setiap transaksi dapat dikendalikan dengan baik. Hubungan kepemilikan ini bersifat eksklusif, sehingga kelas MoneyStorage tidak akan bisa bekerja jika tidak ada kelas MoneyStorageHandler, dan begitu pula sebaliknya. Ibarat sebuah sistem yang tidak memiliki pengendali di belakangnya, maka sistem tersebut tidak akan berfungsi, dan jika hanya ada sistem tanpa

objek yang akan dikendalikan, pasti tidak akan terjadi apa-apa. Oleh karena itulah, hubungan kedua kelas ini bersifat komposisi (asosiasi kepemilikan eksklusif). Satu penyimpan uang dikendalikan oleh tepat satu pengendali, dan satu pengendali tepat mengendalikan satu ruang penyimpanan uang.

Sementara itu, kelas MoneyStorageHandler sendiri memiliki relasi berupa **generalization** dengan interface Runnable. Hal ini menandakan bahwa terjadi inheritance dari interface Runnable ke kelas MoneyStorageHandler. Sifat yang diturunkan adalah method run(), yang menunjukkan bahwa kelas ini akan menerapkan multithreading dalam implementasinya.

Dengan demikian, secara keseluruhan, terdapat tiga relasi yang terbentuk pada hubungan antarkelas di program aplikasi ATM Simulator ini, yaitu:

- Association antara kelas ATM dengan kelas MoneyStorage.
- Composition antara kelas MoneyStorage dengan kelas MoneyStorageHandler.
- Generalization antara kelas MoneyStorageHandler dengan interface Runnable.

## D. Multithreading

Multithreading adalah teknik pemrograman yang memungkinkan beberapa subproses dalam program dapat berjalan secara paralel. Sebuah thread adalah sebuah bagian program yang dapat berjalan mandiri, sehingga dua atau lebih thread dapat berjalan bersamaan, tanpa yang satu harus menunggu selesainya yang lain. Dalam program aplikasi ATM Simulator ini, poin multithreading terdapat di fitur kedua program, yaitu ketika aplikasi memantau transaksi yang terjadi di setiap ATM.

Proses multithreading dalam aplikasi ini berlangsung dengan membuat empat buah thread, di mana masing-masing thread mewakili setiap objek mesin ATM yang terdapat di bank tersebut. Thread-thread tersebut mulai dideklarasikan ketika administrator menekan angka 2 pada menu utama. Thread baru akan dijalankan ketika mesin ATM diketahui dalam kondisi yang baik. Lalu, method start akan memicu dijalankannya method run di kelas MoneyStorageHandler yang sudah mewariskan method run() pada interface Runnable, yang menandakan bahwa keempat thread tersebut sudah dijalankan secara bersamaan.

Catatan penting lain yang perlu diingat adalah fungsi main() pada kelas Main juga merupakan suatu thread yang sedang berjalan. Untuk mencegah fungsi main() berbentrokan waktu atau selesai lebih dahulu daripada thread-thread ATM tersebut, kita perlu memanggil method join() untuk setiap thread ATM, sehingga thread ATM akan didahulukan sampai selesai, lalu thread fungsi main() dilanjutkan kembali. Untuk lebih jelasnya, deskripsi lebih detail akan ditunjukkan pada analisis source code.

## E. Source Code Analysis

Berikut ini disajikan source code beserta penjelasan dan deskripsi-analisis dari setiap kelas pada program aplikasi ATM Simulator Bank Fentira.

### *(1) Main Class (Main.java)*

```
1  import java.util.Scanner;
2
3  public class Main {
4
5      public static void main(String[] args) {
6
7          Scanner scan = new Scanner(System.in);
8
9          MoneyStorage ms = new MoneyStorage();
10         MoneyStorageHandler msh = new MoneyStorageHandler(ms);
11
12         int choice;
13
14         do {
15             System.out.println("=================================================");
16             System.out.println("|| Selamat datang di Simulator ATM Bank Fentira ||");
17             System.out.println("=================================================");
18             System.out.println("Pilih salah satu tindakan berikut:");
19             System.out.println("1. Masukkan uang ke storage ATM");
20             System.out.println("2. Pantau penarikan setiap mesin ATM");
21             System.out.println("3. Kosongkan uang dari storage ATM");
22             System.out.println("4. Tampilkan log transaksi setiap mesin ATM");
23             System.out.println("5. Hapus log transaksi semua mesin ATM");
24             System.out.println("6. Keluar dari program simulasi");
25             System.out.print(">> ");
26
27             choice = scan.nextInt();
28             System.out.println();
29
30             if(choice == 1) {
31                 int total;
32
33                 do {
34                     System.out.print("Masukkan jumlah uang ke mesin ATM: ");
35                     total = scan.nextInt();
36
37                     if(total % 100000 != 0) {
38                         System.out.println("Masukkan nominal pecahan Rp100.000,- saja. Coba lagi.");
39                         System.out.println();
40                     }
41                 } while(total % 100000 != 0);
42
43                 ms.setNominal(ms.getNominal() + total);
44                 System.out.println("Sejumlah uang dengan nominal Rp" + ms.getNominal() + ",- siap ditransaksi oleh nasabah.");
45                 System.out.println("Mesin siap digunakan.");
46                 System.out.println();
47             }
```

```
48          else if(choice == 2) {
49              Thread atm1 = new Thread(msh);
50              Thread atm2 = new Thread(msh);
51              Thread atm3 = new Thread(msh);
52              Thread atm4 = new Thread(msh);
53
54              atm1.setName("ATM 1");
55              atm2.setName("ATM 2");
56              atm3.setName("ATM 3");
57              atm4.setName("ATM 4");
58
59              if(msh.a1.isKondisi() == true) {
60                  atm1.start();
61              }
62
63              if(msh.a2.isKondisi() == true) {
64                  atm2.start();
65              }
66
67              if(msh.a3.isKondisi() == true) {
68                  atm3.start();
69              }
70
71              if(msh.a4.isKondisi() == true) {
72                  atm4.start();
73              }
74
75              try {
76                  atm1.join();
77              }
78              catch (InterruptedException e) {
79                  System.out.println("[X] Mesin ATM 1 mengalami kegagalan transaksi.");
80              }
81
82              try {
83                  atm2.join();
84              }
85              catch (InterruptedException e) {
86                  System.out.println("[X] Mesin ATM 2 mengalami kegagalan transaksi.");
87              }
88
89              try {
90                  atm3.join();
91              }
92              catch (InterruptedException e) {
93                  System.out.println("[X] Mesin ATM 3 mengalami kegagalan transaksi.");
94              }
95
96              try {
97                  atm4.join();
98              }
99              catch (InterruptedException e) {
100                 System.out.println("[X] Mesin ATM 4 mengalami kegagalan transaksi.");
101             }
101             }
102         }
103         else if(choice == 3) {
104             if(ms.getNominal() == 0) {
105                 System.out.println("Mesin sudah kosong.");
106                 System.out.println("Saat ini mesin masih belum bisa dipakai sampai sejumlah uang dimasukkan.");
107             }
108             else {
109                 ms.setNominal(0);
110                 System.out.println("Mesin berhasil dikosongkan.");
111                 System.out.println("Nasabah belum bisa melakukan penarikan sampai uang tersedia di dalam mesin.");
112             }
113             System.out.println();
114         }
115         else if(choice == 4) {
116             System.out.println("Log transaksi ATM 1:");
117             if(msh.a1.getLogTransaksi().isEmpty()) {
118                 System.out.println(">> ATM 1 belum melakukan transaksi apapun sejauh ini.");
119             }
120             else {
121                 for(int i=0; i<msh.a1.getLogTransaksi().size(); i++) {
122                     System.out.println(">> " + (i+1) + ". Mesin ATM 1 melakukan transaksi sebesar Rp" +
123                     msh.a1.getLogTransaksi().get(i) + ",-");
124                 }
125             }
126             System.out.println();
```

```
127
128                 System.out.println("Log transaksi ATM 2:");
129                 if(msh.a2.getLogTransaksi().isEmpty()) {
130                     System.out.println(">> ATM 2 belum melakukan transaksi apapun sejauh ini.");
131                 }
132                 else {
133                     for(int i=0; i<msh.a2.getLogTransaksi().size(); i++) {
134                         System.out.println(">> " + (i+1) + ". Mesin ATM 2 melakukan transaksi sebesar Rp" +
135                         msh.a2.getLogTransaksi().get(i) + ",-");
136                     }
137                 }
138                 System.out.println();
139
140                 System.out.println("Log transaksi ATM 3:");
141                 if(msh.a3.getLogTransaksi().isEmpty()) {
142                     System.out.println(">> ATM 3 belum melakukan transaksi apapun sejauh ini.");
143                 }
144                 else {
145                     for(int i=0; i<msh.a3.getLogTransaksi().size(); i++) {
146                         System.out.println(">> " + (i+1) + ". Mesin ATM 3 melakukan transaksi sebesar Rp" +
147                         msh.a3.getLogTransaksi().get(i) + ",-");
148                     }
149                 }
150                 System.out.println();
151
152                 System.out.println("Log transaksi ATM 4:");
153                 if(msh.a4.getLogTransaksi().isEmpty()) {
154                     System.out.println(">> ATM 4 belum melakukan transaksi apapun sejauh ini.");
155                 }
156                 else {
157                     for(int i=0; i<msh.a4.getLogTransaksi().size(); i++) {
158                         System.out.println(">> " + (i+1) + ". Mesin ATM 4 melakukan transaksi sebesar Rp" +
159                         msh.a4.getLogTransaksi().get(i) + ",-");
160                     }
161                 }
162                 System.out.println();
163
164                 System.out.println("Total transaksi di mesin ATM 1 = Rp" + msh.a1.getTotal() + ",-");
165                 System.out.println("Total transaksi di mesin ATM 2 = Rp" + msh.a2.getTotal() + ",-");
166                 System.out.println("Total transaksi di mesin ATM 3 = Rp" + msh.a3.getTotal() + ",-");
167                 System.out.println("Total transaksi di mesin ATM 4 = Rp" + msh.a4.getTotal() + ",-");
168                 System.out.println();
169             }
170             else if(choice == 5) {
171                 if(msh.a1.getLogTransaksi().isEmpty() && msh.a2.getLogTransaksi().isEmpty() &&
172                 msh.a3.getLogTransaksi().isEmpty() && msh.a4.getLogTransaksi().isEmpty()) {
173                     System.out.println("Log transaksi keempat mesin ATM masih kosong.");
174                     System.out.println("Tidak ada data yang perlu dihapus.");
175                     System.out.println();
176                 }
177                 else {
178                     scan.nextLine();
179                     char confirm;
180
181                     System.out.println("Apakah Anda yakin ingin menghapus semua log transaksi?");
182                     System.out.println("Anda tidak dapat mengembalikannya setelah terhapus.");
183                     do {
184                         System.out.print("Konfirmasi dengan menjawab [Y atau N]: ");
185                         confirm = scan.nextLine().charAt(0);
186                     } while(confirm != 'Y' && confirm != 'N');
187
188                     if(confirm == 'Y') {
189                         msh.a1.getLogTransaksi().clear();
190                         msh.a2.getLogTransaksi().clear();
191                         msh.a3.getLogTransaksi().clear();
192                         msh.a4.getLogTransaksi().clear();
193
194                         msh.a1.setTotal(0);
195                         msh.a2.setTotal(0);
196                         msh.a3.setTotal(0);
197                         msh.a4.setTotal(0);
198
199                         System.out.println("Log transaksi semua mesin ATM berhasil dihapus.");
200                         System.out.println();
201                     }
202                     else {
203                         System.out.println("Log transaksi batal dihapus.");
204                         System.out.println();
205                     }
206                 }
207             }
208             else if(choice == 6) {
209                 System.out.println("Terima kasih telah menggunakan simulator ATM Bank Fentira.");
210             }
211             else {
212                 System.out.println("Maaf, Anda memasukkan input yang salah.");
213                 System.out.println("Coba lagi.");
214                 System.out.println();
215             }
216
217             System.out.println();
218
219         } while(choice != 6);
220
221     }
222 }
```

Secara umum, source code pada Main class ini sudah dijelaskan cukup detail pada bagian penjelasan fitur aplikasi, hanya saja penjelasan tersebut diolah ke dalam bentuk codingan Java. Sebuah menu dibuat untuk menjalankan kelima fitur yang disediakan. Sebelumnya, objek storage dan handler-nya harus dibentuk terlebih dahulu untuk mengatur mesin-mesin ATM yang ada.

Fitur pertama akan memasukkan nominal tertentu ke dalam storage, dengan validasi bahwa uang yang dimasukkan hanya pecahan Rp100.000,- saja. Dengan menggunakan method setNominal(), jumlah uang di dalam storage akan bertambah sejumlah yang dimasukkan oleh administrator.

Fitur kedua adalah fitur yang memanfaatkan multithreading. Dapat kita perhatikan secara saksama, bahwa thread ATM 1 sampai 4 dideklarasikan di sini. Dengan memvalidasi kondisi mesin, method start() dipanggil untuk memulai proses multithreading untuk keempat mesin. Pada posisi ini, keempat thread sedang berjalan bersamaan secara paralel melakukan penarikan tunai pada storage. Method join() digunakan untuk memastikan bahwa thread utama di fungsi main() baru akan dilanjutkan ketika keempat thread ATM ini selesai dieksekusi.

Fitur ketiga hanya melawan kerja fitur pertama, sehingga baris kode yang diperlukan hanya memanggil method setNominal() dengan argumen nilai nol (0) agar storage kembali kosong.

Fitur keempat memanfaatkan atribut berupa ArrayList suatu Integer pada kelas ATM. ArrayList ini memang diperuntukkan bagi log transaksi yang dilakukan oleh setiap mesin. Untuk melihat semua log transaksi, kita hanya perlu melakukan looping sebanyak size dari ArrayList tersebut, lalu mencetak semua nilai nominal yang ditransaksi dari waktu ke waktu. Nominal total juga dapat dicetak pada fitur ini melalui atribut nominal pada kelas ATM.

Fitur kelima sebenarnya hanya melakukan delete data pada ArrayList yang sudah dibuat. Akan tetapi, diperlukan validasi untuk memastikan bahwa data benar-benar ingin dihapus, sehingga diperlukan looping selama input konfirmasi bukan huruf yang menjadi ekspektasi jawaban, yaitu ‘Y’ dan ‘N’.

### *(2) ATM Class (ATM.java)*

```java
import java.util.ArrayList;

public class ATM {
    private boolean kondisi;
    private String nama;
    private int total = 0;
    private ArrayList<Integer> logTransaksi = new ArrayList<Integer>();

    public ATM(boolean kondisi, String nama) {
        super();
        this.kondisi = kondisi;
        this.nama = nama;
    }

    public boolean isKondisi() {
        return kondisi;
    }

    public void setKondisi(boolean kondisi) {
        this.kondisi = kondisi;
    }

    public String getNama() {
        return nama;
    }


    public void setNama(String nama) {
        this.nama = nama;
    }

    public int getTotal() {
        return total;
    }

    public void setTotal(int total) {
        this.total = total;
    }

    public ArrayList<Integer> getLogTransaksi() {
        return logTransaksi;
    }

    public void setLogTransaksi(int total) {
        logTransaksi.add(total);
    }

}
```

Pada kelas ATM ini, pertama-tama kita perlu mendeklarasikan semua atribut pada objek mesin ATM ini, mulai dari kondisi kelayakan, nama mesin ATM, total nominal transaksi, dan riwayat transaksi dalam bentuk ArrayList. Kemudian, dibuatlah sebuah constructor yang akan menjadi instance pendeklarasian kelas ATM. Dilanjutkan dengan proses enkapsulasi dengan menambahkan getter dan setter pada setiap atribut yang dilindungi. Pada tahap ini, kelas ATM sudah jadi dan siap untuk membuat sebuah objek ATM.

### *(3) MoneyStorage Class (MoneyStorage.java)*

```java

public class MoneyStorage {
    private int nominal = 0;

    public int getNominal() {
        return nominal;
    }

    public void setNominal(int nominal) {
        this.nominal = nominal;
    }

    public void penarikan(int total) {
        nominal -= total;
    }
}

```

Hampir sama dengan kelas ATM, kelas MoneyStorage juga perlu mendeklarasikan atributnya, yang dalam hal ini hanya atribut nominal yang tersimpan di dalam storage. Setelah itu, atribut ini dienkapsulasi dengan membuatkan sebuah getter dan setter. Sebagai tambahan, pada kelas ini ditambahkan juga sebuah method penarikan() untuk mengurangi nilai nominal pada storage jika terjadi transaksi penarikan uang.

*(4) MoneyStorageHandler Class (MoneyStorageHandler.java)*

```
1
2  public class MoneyStorageHandler implements Runnable {
3      private MoneyStorage ms;
4      ATM a1 = new ATM(true, "ATM 1");
5      ATM a2 = new ATM(true, "ATM 2");
6      ATM a3 = new ATM(true, "ATM 3");
7      ATM a4 = new ATM(true, "ATM 4");
8
9      public MoneyStorageHandler(MoneyStorage ms) {
10         super();
11         this.ms = ms;
12     }
13
14     @Override
15     public void run() {
16         do {
17             tarikTunai(100000);
18             try {
19                 Thread.sleep(1500);
20             }
21             catch (InterruptedException e) {
22                 System.out.println("Transaksi gagal!");
23             }
24         } while(ms.getNominal() > 0);
25     }
26
27     private synchronized void tarikTunai(int totalPenarikan) {
28         if (ms.getNominal() >= totalPenarikan) {
29             System.out.println("[?] " + Thread.currentThread().getName() + " sedang melakukan penarikan Rp" + totalPenarikan);
30             ms.penarikan(totalPenarikan);
31             System.out.println("[V] " + Thread.currentThread().getName() + " selesai melakukan penarikan sejumlah Rp"
32             + totalPenarikan);
33             System.out.println();
34
35             if(Thread.currentThread().getName().equals("ATM 1")) {
36                 a1.setLogTransaksi(100000);
37                 a1.setTotal(a1.getTotal() + 100000);
38             }
39             else if(Thread.currentThread().getName().equals("ATM 2")) {
40                 a2.setLogTransaksi(100000);
41                 a2.setTotal(a2.getTotal() + 100000);
42             }
43             else if(Thread.currentThread().getName().equals("ATM 3")) {
44                 a3.setLogTransaksi(100000);
45                 a3.setTotal(a3.getTotal() + 100000);
46             }
47             else if(Thread.currentThread().getName().equals("ATM 4")) {
48                 a4.setLogTransaksi(100000);
49                 a4.setTotal(a4.getTotal() + 100000);
50             }
51         }
52         else {
53             System.out.println("[!] Uang tidak tersedia lagi untuk " + Thread.currentThread().getName());
54         }
55     }
56 }
57
```

Di kelas ini, objek yang diharapkan adalah suatu sistem pengendali storage yang dapat memengaruhi kinerja keempat mesin ATM, sehingga kita perlu mendeklarasikan sebuah objek dari kelas MoneyStorage, dan empat buah ATM dari kelas ATM. Struktur ini dibuat seperti halnya realita pada skenario yang digunakan.

Selanjutnya, untuk mengimplementasikan multithreading pada Main class, perlu dideklarasikan sebuah method bernama run() yang diwariskan dari interface Runnable. Method ini berisi setiap task yang harus dilakukan oleh thread, dalam hal ini ATM, yaitu melakukan transaksi penarikan tunai sebesar Rp100.000,-. Keempat ATM akan melakukan method tarikTunai() bersamaan, lalu keempat thread diistirahatkan 1,5 detik, dan dilanjutkan terus-menerus sampai nominal di dalam storage habis.

Untuk method tarikTunai() sendiri, terdapat beberapa konfigurasi yang dilakukan. Ketika sebuah ATM, dalam hal ini thread, melakukan penarikan tunai, sistem akan memulai proses penarikan, lalu memanggil method penarikan() pada objek dari kelas MoneyStorage. Method penarikan() ini memicu terjadinya pengurangan jumlah uang di dalam storage. Akibatnya, setiap uang yang ditransaksikan oleh thread akan dicatat di dalam ArrayList yang merupakan atribut dari ATM itu sendiri. Dengan menggunakan method setLogTransaksi(), sebuah value ditambahkan sebagai sebuah elemen dalam ArrayList logTransaksi. Kemudian atribut nominal juga ditambahkan dari kondisi awalnya. Jadilah mesin ATM dengan atribut yang sepenuhnya dikendalikan oleh kelas MoneyStorageHandler.

## F. Output Screenshots

Berikut ini kami sajikan hasil tangkapan layar output console dari program aplikasi ATM Simulator yang telah dibuat. Tangkapan layar akan diberikan untuk semua fitur yang terlibat dalam aplikasi ATM Simulator ini, termasuk error handling ketika ada kemungkinan kesalahan input.

- Fitur 1

```
==================================================
|| Selamat datang di Simulator ATM Bank Fentira ||
==================================================
Pilih salah satu tindakan berikut:
1. Masukkan uang ke storage ATM
2. Pantau penarikan setiap mesin ATM
3. Kosongkan uang dari storage ATM
4. Tampilkan log transaksi setiap mesin ATM
5. Hapus log transaksi semua mesin ATM
6. Keluar dari program simulasi
>> 1

Masukkan jumlah uang ke mesin ATM: 500000
Sejumlah uang dengan nominal Rp500000,- siap ditransaksi oleh nasabah.
Mesin siap digunakan.
```

- Fitur 2

```
==================================================
|| Selamat datang di Simulator ATM Bank Fentira ||
==================================================
Pilih salah satu tindakan berikut:
1. Masukkan uang ke storage ATM
2. Pantau penarikan setiap mesin ATM
3. Kosongkan uang dari storage ATM
4. Tampilkan log transaksi setiap mesin ATM
5. Hapus log transaksi semua mesin ATM
6. Keluar dari program simulasi
>> 2

[?] ATM 1 sedang melakukan penarikan Rp100000
[V] ATM 1 selesai melakukan penarikan sejumlah Rp100000

[?] ATM 4 sedang melakukan penarikan Rp100000
[V] ATM 4 selesai melakukan penarikan sejumlah Rp100000

[?] ATM 2 sedang melakukan penarikan Rp100000
[V] ATM 2 selesai melakukan penarikan sejumlah Rp100000

[?] ATM 3 sedang melakukan penarikan Rp100000
[V] ATM 3 selesai melakukan penarikan sejumlah Rp100000


[?] ATM 1 sedang melakukan penarikan Rp100000
[V] ATM 1 selesai melakukan penarikan sejumlah Rp100000

[!] Uang tidak tersedia lagi untuk ATM 4
[!] Uang tidak tersedia lagi untuk ATM 2
[!] Uang tidak tersedia lagi untuk ATM 3
```

- Fitur 3

Jika terdapat uang di dalam storage:

```
==================================================
|| Selamat datang di Simulator ATM Bank Fentira ||
==================================================
Pilih salah satu tindakan berikut:
1. Masukkan uang ke storage ATM
2. Pantau penarikan setiap mesin ATM
3. Kosongkan uang dari storage ATM
4. Tampilkan log transaksi setiap mesin ATM
5. Hapus log transaksi semua mesin ATM
6. Keluar dari program simulasi
>> 3

Mesin berhasil dikosongkan.
Nasabah belum bisa melakukan penarikan sampai uang tersedia di dalam mesin.
```

Jika tidak ada uang di dalam storage:

```
==================================================
|| Selamat datang di Simulator ATM Bank Fentira ||
==================================================
Pilih salah satu tindakan berikut:
1. Masukkan uang ke storage ATM
2. Pantau penarikan setiap mesin ATM
3. Kosongkan uang dari storage ATM
4. Tampilkan log transaksi setiap mesin ATM
5. Hapus log transaksi semua mesin ATM
6. Keluar dari program simulasi
>> 3

Mesin sudah kosong.
Saat ini mesin masih belum bisa dipakai sampai sejumlah uang dimasukkan.
```

- Fitur 4

```
==================================================
|| Selamat datang di Simulator ATM Bank Fentira ||
==================================================
Pilih salah satu tindakan berikut:
1. Masukkan uang ke storage ATM
2. Pantau penarikan setiap mesin ATM
3. Kosongkan uang dari storage ATM
4. Tampilkan log transaksi setiap mesin ATM
5. Hapus log transaksi semua mesin ATM
6. Keluar dari program simulasi
>> 4

Log transaksi ATM 1:
>> 1. Mesin ATM 1 melakukan transaksi sebesar Rp100000,-
>> 2. Mesin ATM 1 melakukan transaksi sebesar Rp100000,-

Log transaksi ATM 2:
>> 1. Mesin ATM 2 melakukan transaksi sebesar Rp100000,-

Log transaksi ATM 3:
>> 1. Mesin ATM 3 melakukan transaksi sebesar Rp100000,-

Log transaksi ATM 4:
>> 1. Mesin ATM 4 melakukan transaksi sebesar Rp100000,-


Total transaksi di mesin ATM 1 = Rp200000,-
Total transaksi di mesin ATM 2 = Rp100000,-
Total transaksi di mesin ATM 3 = Rp100000,-
Total transaksi di mesin ATM 4 = Rp100000,-
```

- Fitur 5

Jika memilih ‘N’:

```
==================================================
|| Selamat datang di Simulator ATM Bank Fentira ||
==================================================
Pilih salah satu tindakan berikut:
1. Masukkan uang ke storage ATM
2. Pantau penarikan setiap mesin ATM
3. Kosongkan uang dari storage ATM
4. Tampilkan log transaksi setiap mesin ATM
5. Hapus log transaksi semua mesin ATM
6. Keluar dari program simulasi
>> 5

Apakah Anda yakin ingin menghapus semua log transaksi?
Anda tidak dapat mengembalikannya setelah terhapus.
Konfirmasi dengan menjawab [Y atau N]: N
Log transaksi batal dihapus.
```

Jika memilih ‘Y’:

```
==================================================
|| Selamat datang di Simulator ATM Bank Fentira ||
==================================================
Pilih salah satu tindakan berikut:
1. Masukkan uang ke storage ATM
2. Pantau penarikan setiap mesin ATM
3. Kosongkan uang dari storage ATM
4. Tampilkan log transaksi setiap mesin ATM
5. Hapus log transaksi semua mesin ATM
6. Keluar dari program simulasi
>> 5

Apakah Anda yakin ingin menghapus semua log transaksi?
Anda tidak dapat mengembalikannya setelah terhapus.
Konfirmasi dengan menjawab [Y atau N]: Y
Log transaksi semua mesin ATM berhasil dihapus.
```

- Exit

```
==================================================
|| Selamat datang di Simulator ATM Bank Fentira ||
==================================================
Pilih salah satu tindakan berikut:
1. Masukkan uang ke storage ATM
2. Pantau penarikan setiap mesin ATM
3. Kosongkan uang dari storage ATM
4. Tampilkan log transaksi setiap mesin ATM
5. Hapus log transaksi semua mesin ATM
6. Keluar dari program simulasi
>> 6

Terima kasih telah menggunakan simulator ATM Bank Fentira.
```

- Kesalahan input

```
==================================================
|| Selamat datang di Simulator ATM Bank Fentira ||
==================================================
Pilih salah satu tindakan berikut:
1. Masukkan uang ke storage ATM
2. Pantau penarikan setiap mesin ATM
3. Kosongkan uang dari storage ATM
4. Tampilkan log transaksi setiap mesin ATM
5. Hapus log transaksi semua mesin ATM
6. Keluar dari program simulasi
>> 7

Maaf, Anda memasukkan input yang salah.
Coba lagi.
```