

OBJECT ORIENTED PROGRAMMING

Fendy Wijaya - 2602092150

Tiara Intan Kusuma - 2602172220



Project Descriptions

ATM Simulator



01

Fendy adalah pebisnis muda dan baru saja mendirikan Bank Fentira. Baginya, kenyamanan dan keamanan nasabah merupakan prioritas utama.

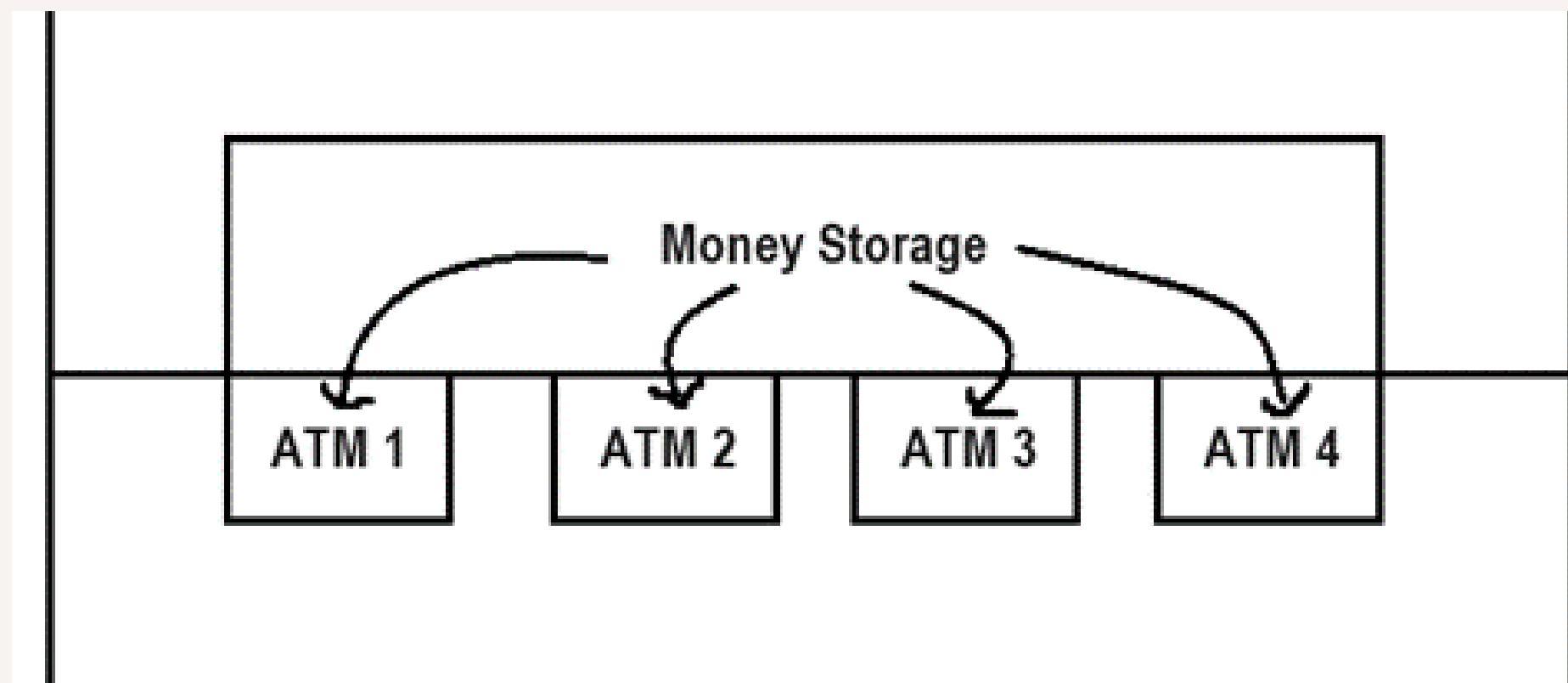
02

Data CNN tahun 2015 menunjukkan bahwa 1/3 kasus penipuan ATM dunia terjadi di Indonesia.

03

Fendy ingin membuat program yang dapat memantau perputaran uang di mesin ATM Bank Fentira.

Fendy telah membeli 4 buah mesin ATM yang akan diletakkan di bagian depan banknya. Mesin ATM ini tidak memiliki penyimpanan uang sendiri melainkan keempat mesin mengakses tempat penyimpanan uang yang sama.



Program simulator mencatat setiap detail :

- kondisi mesin
- jumlah uang dalam penyimpanan
- riwayat transaksi tiap mesin ATM.





Application Features ➔

Masukkan uang ke storage ATM

Simulator akan otomatis menghitung jumlah uang pada storage ATM. Fitur ini menambah sejumlah uang sesuai perintah administrator. Untuk alasan konsistensi, mesin hanya dapat menerima pecahan Rp 100.000,-

Kosongkan uang dari storage ATM

Fitur ini hanya bagi administrator yang ingin menarik seluruh uangnya. Apabila terjadi perampokan atau kesalahan perhitungan uang, fitur ini akan menarik kembali seluruh uang sehingga uang di dalam storage menjadi Rp 0,-

Pantau penarikan setiap mesin ATM

Setiap transaksi akan di-print di console, mulai saat mesin memproses hingga selesai mengeluarkan uang. Apabila uang dalam storage habis, maka seluruh mesin mengeluarkan notifikasi terkait hal ini.

Tampilkan transaksi tiap mesin ATM

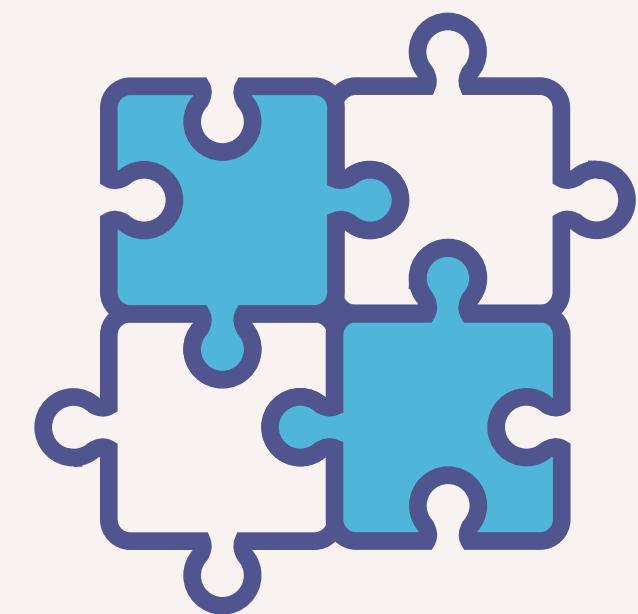
Setiap riwayat transaksi keempat mesin akan dicatat dan ditampilkan oleh aplikasi. Data fitur ini akan melacak tiap transaksi untuk mengetahui frekuensi transaksi tiap mesin.





Hapus transaksi semua mesin ATM

Untuk alasan efisiensi memori, administrator dapat menghapus log transaksi jika sudah tidak diperlukan. Penghapusan data memerlukan validasi untuk mencegah ketidaksengajaan penghapusan data.



Keluar dari program simulasi

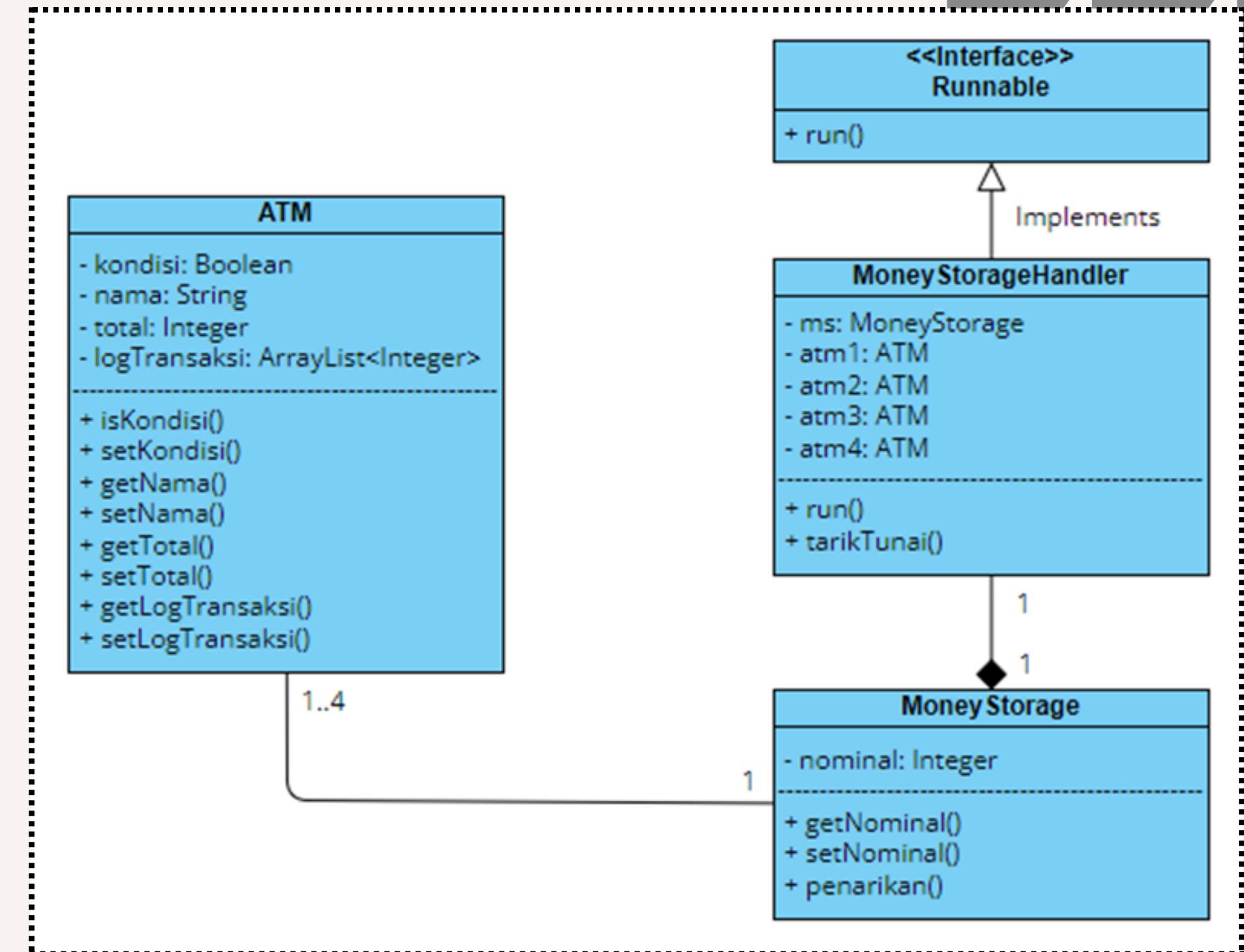
Untuk keluar dari loopingan program simulasi yang sedang berjalan, administrator dapat menekan tombol '6' untuk menghentikan aplikasi simulator ATM.



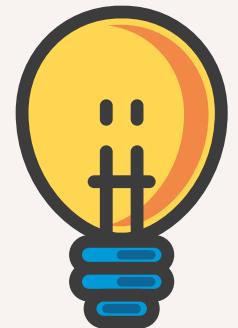
Class Relationships

Terdapat 4 kelas :

- Main
- ATM
- MoneyStorage
- MoneyStorageHandler



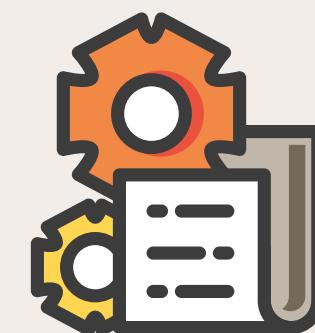
RELASI ANTAR KELAS



ATM - MoneyStorage
memiliki hubungan *association*



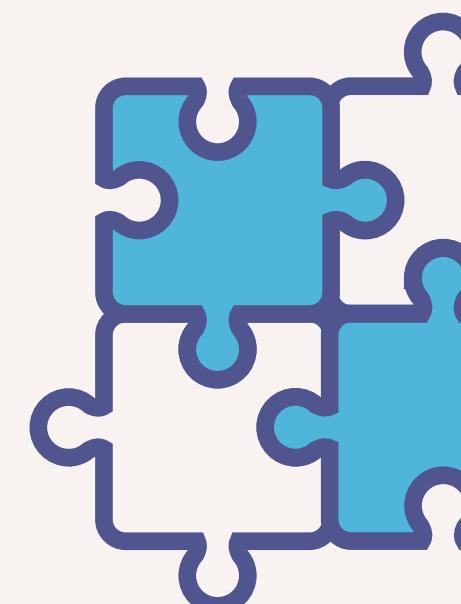
MoneyStorage - MoneyStorageHandler
memiliki hubungan *composition*



MoneyStorageHandler - Runnable
memiliki relasi *generalization (inheritance)*



Multithreading



Multithreading adalah teknik pemrograman yang memungkinkan beberapa subproses dalam program dapat berjalan secara paralel. Dalam program aplikasi ATM Simulator ini, poin multithreading terdapat di fitur kedua program, yaitu ketika aplikasi memantau transaksi yang terjadi di setiap ATM.

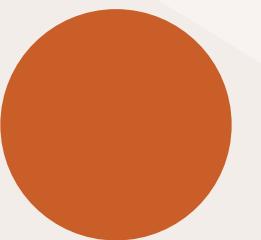


Proses multithreading berlangsung dengan membuat empat buah thread yang mewakili setiap mesin ATM. Thread-thread tersebut mulai dideklarasikan ketika administrator menekan 2 pada menu utama. Thread baru akan dijalankan ketika mesin ATM diketahui dalam kondisi baik. Lalu, method start akan memicu dijalankannya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





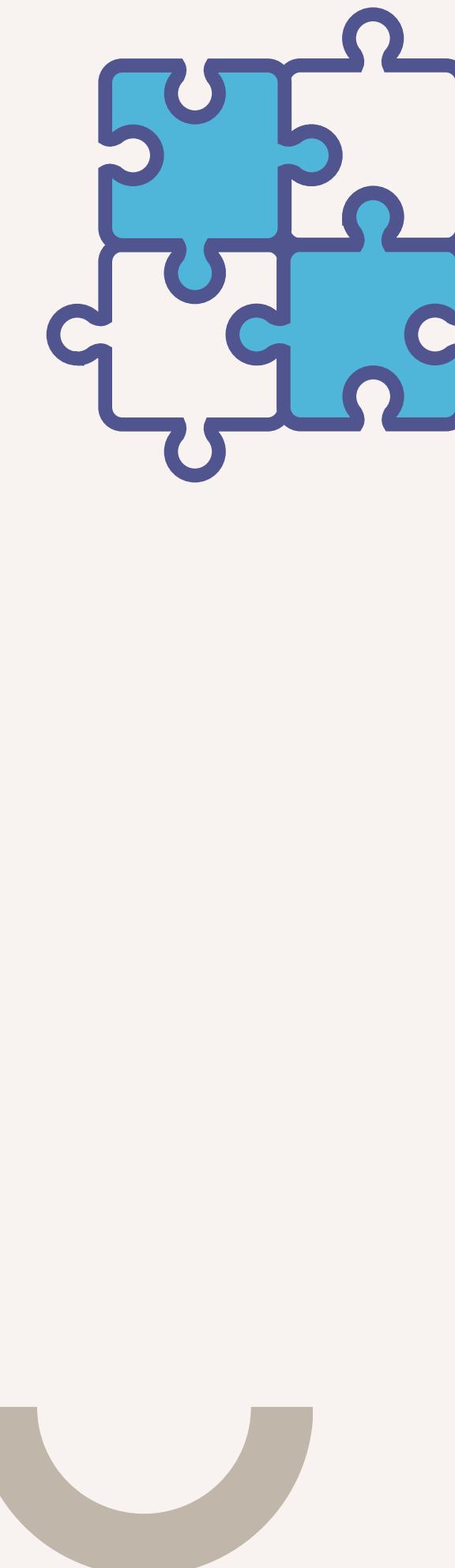
Source Code Analysis



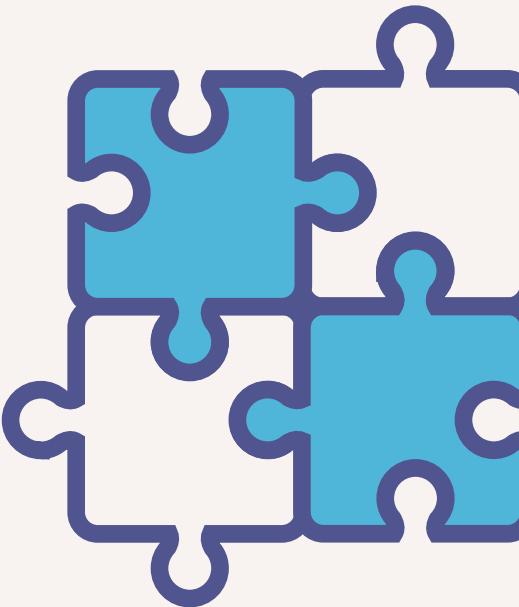
Main class (Main.java)

```
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        }
49    }
50}
```

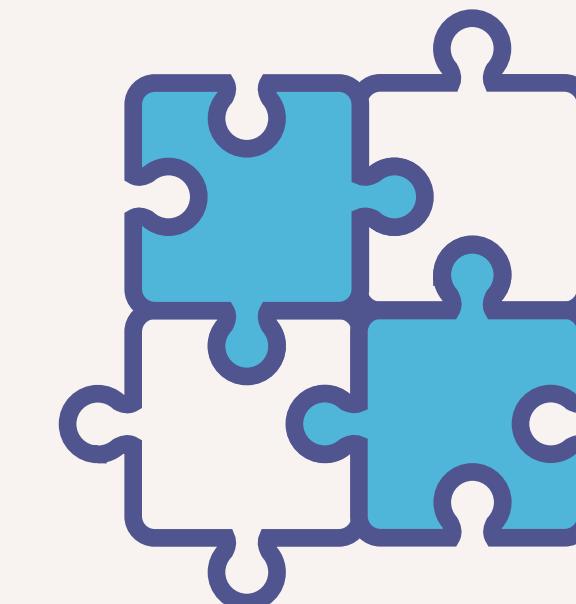
```
48     else if(choice == 2) {
49         Thread atm1 = new Thread(msh);
50         Thread atm2 = new Thread(msh);
51         Thread atm3 = new Thread(msh);
52         Thread atm4 = new Thread(msh);
53
54         atm1.setName("ATM 1");
55         atm2.setName("ATM 2");
56         atm3.setName("ATM 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        }
```

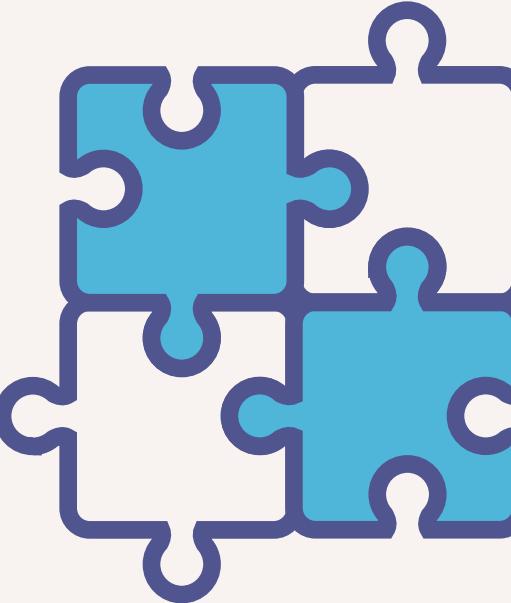


```
101     }
102 }
103 else if(choice == 3) {
104     if(ms.getNominal() == 0) {
105         System.out.println("Mesin sudah kosong.");
106         System.out.println("Saat ini mesin masih belum bisa dipakai sampai sejumlah uang dimasukkan.");
107     }
108 else {
109     ms.setNominal(0);
110     System.out.println("Mesin 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 }
```





```
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 }
```



Fitur 1 :

Dengan method setNominal(), jumlah uang di dalam storage akan bertambah sesuai yang dimasukkan oleh administrator



Fitur 2 :

Dengan memvalidasi kondisi mesin, method start() dipanggil untuk memulai proses multithreading. Method join() memastikan bahwa thread main baru akan dilanjutkan ketika keempat thread selesai dieksekusi.





Fitur 3:

Baris-baris kode hanya memanggil method setNominal() dengan argumen nilai nol (0) agar storage kembali kosong.



Fitur 4:

Fitur ini memanfaatkan ArrayList suatu Integer pada kelas ATM. ArrayList diperuntukkan bagi log transaksi yang dilakukan setiap mesin. Untuk melihat semua log, kita looping sebanyak size dari ArrayList, lalu mencetak semua transaksi.





Fitur 5 :

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'.



ATM Class (ATM.java)

```
1 import java.util.ArrayList;
2
3 public class ATM {
4     private boolean kondisi;
5     private String nama;
6     private int total = 0;
7     private ArrayList<Integer> logTransaksi = new ArrayList<Integer>();
8
9     public ATM(boolean kondisi, String nama) {
10         super();
11         this.kondisi = kondisi;
12         this.nama = nama;
13     }
14
15     public boolean isKondisi() {
16         return kondisi;
17     }
18
19     public void setKondisi(boolean kondisi) {
20         this.kondisi = kondisi;
21     }
22
23     public String getNama() {
24         return nama;
25     }
26
27     public void setNama(String nama) {
28         this.nama = nama;
29     }
30
31     public int getTotal() {
32         return total;
33     }
34
35     public void setTotal(int total) {
36         this.total = total;
37     }
38
39     public ArrayList<Integer> getLogTransaksi() {
40         return logTransaksi;
41     }
42
43     public void setLogTransaksi(int total) {
44         logTransaksi.add(total);
45     }
46
47 }
```

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.

Money Storage Class (MoneyStorage.java)

```
1  public class MoneyStorage {
2      private int nominal = 0;
3
4
5      public int getNominal() {
6          return nominal;
7      }
8
9      public void setNominal(int nominal) {
10         this.nominal = nominal;
11     }
12
13     public void penarikan(int total) {
14         nominal -= total;
15     }
16 }
17
```

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.

MoneyStorageHandler Class (MoneyStorageHandler.java)

```
1  public class MoneyStorageHandler implements Runnable {
2      private MoneyStorage ms;
3      ATM a1 = new ATM(true, "ATM 1");
4      ATM a2 = new ATM(true, "ATM 2");
5      ATM a3 = new ATM(true, "ATM 3");
6      ATM a4 = new ATM(true, "ATM 4");
7
8      public MoneyStorageHandler(MoneyStorage ms) {
9          super();
10         this.ms = ms;
11     }
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 }
```

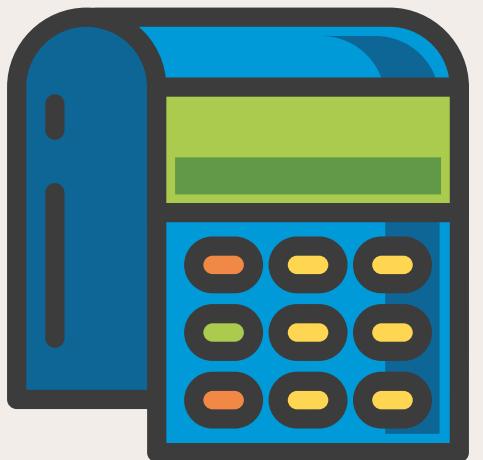
Di kelas ini, objek yang diharapkan adalah suatu sistem pengendali storage yang dapat memengaruhi kinerja keempat mesin ATM, sehingga kita perlu mendeklarasikan sebuah objek dari kelas MoneyStorage, dan empat buah ATM dari kelas ATM.

Selanjutnya, 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.

OUTPUT SCREENSHOTS



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 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 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.



Terima Kasih

