LAPORAN PRAKTIKUM

ALGORITMA DAN PEMROGRAMAN

IMPLEMENTASI METHODE SETTER & GETTER DALAM PEMBUATAN PROGRAM REGISTRASI KREDENSIAL

Disusun Oleh:

Haliya Isma Husna Putri Ahmadi
2511532002

Dosen Pengampu:
Wahyudi. Dr. S.T.M.T

Asisten Praktikum:
Muhammad Zaky Al Hafiz

DEPARTEMEN INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

2025

KATA PENGANTAR

Puji dan Syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa karena atas Rahmat dan karunia-Nya, penulis dapat menyelesaikan laporan tugas Praktikum Algoritma dan Pemrograman dengan judul "Implementasi *Methode Setter & Getter* pada Program Registrasi Kredensial" pekan ke-7 tepat pada waktunya.

Penyusunan laporan ini bertujuan untuk memenuhi salah satu tugas dalam mata kuliah Praktikum Algoritma dan Pemrograman, sekaligus sebagai sarana pembelajaran bagi penulis dalam memahami konsep *Object-Oriented Programming (OOP),* khususnya penerapan dua kelas (*class*) serta pemanfaatan *method setter dan getter* dalam pemrograman Java. Melalui praktikum ini, penulis dapat mempelajari bagaimana sebuah program dapat dikelola dengan lebih terstruktur melalui pemisahan *class data* (Akun) dan *class utama* (Registrasi).

Penulis menyadari bahwa laporan ini masih jauh dari sempurna. Oleh karena itu, penulis sangat terbuka terhadap kritik dan saran yang membangun demi perbaikan di masa yang akan datang.

Ucapan terima kasih yang sebesar-besarnya penulis sampaikan kepada dosen pengampu, asisten praktikum, serta semua pihak yang terlibat dalam pelaksanaan praktikum dan penyusunan laporan ini. Semoga laporan ini dapat memberikan manfaat dan menambah wawasan bagi penulis maupun pembaca.

Padang, 14 November 2025

Penulis

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam mempelajari bahasa pemrograman, pemahaman mengenai konsep *Object-Oriented Programming (OOP)* sangat penting karena menjadi dasar dalam pengembangan perangkat lunak yang terstruktur dan mudah dikelola. Salah satu konsep utama dalam *OOP* adalah *encapsulation*, yaitu teknik untuk menyembunyikan data agar tidak dapat diakses secara langsung dari luar kelas. Konsep ini diwujudkan dengan menjadikan variabel bersifat *private* serta menyediakan *method setter* dan *getter* untuk mengatur dan membaca nilai data tersebut secara aman.

Pada praktikum pekan ke-7 ini, mahasiswa mempelajari penerapan dua kelas, yaitu *class data* dan *class utama*, sekaligus memahami bagaimana *setter* dan *getter* digunakan sebagai implementasi dari *encapsulation*. Melalui konsep ini, data dalam program dapat dikelola dengan lebih aman, terstruktur, dan sesuai dengan prinsip dasar *OOP*.

1.2 Tujuan

Tujuan dari pelaksanaan praktikum ini adalah sebagai berikut:

1.2.1 Memahami konsep dasar, logika dan struktur *setter* dan *getter* dalam pengelolaan data program.

1.2.2 Mengimplemantasikan konsep dua kelas yang saling terhubung, di mana satu kelas menyimpan data dan lainnya mengelola *input*.

1.2.3 Membangun logika pemrograman dalam menyusun program dan algoritma yang terstruktur.

1.3 Manfaat

Manfaat yang diharapkan dari pelaksanaan praktikum ini antara lain:

1.3.1 Memberikan pemahaman mengenai konsep *setter* dan *getter*, dalam mengelola dan melindungi data dalam bahasa *Java*.

1.3.2 Memberikan pengalaman praktis dalam mengimplementasikan *setter* dan *getter* dan penerapan dua kelas untuk suatu kasus *real*.

BAB II
PEMBAHASAN

2.1 Uraian Kode Program Java

Berikut merupakan uraian kode program yang dibuat untuk mensimulasikan proses registrasi akun baru, sesuai dengan soal praktikum. Program ini disusun dengan memanfaatkan konsep *Object-Oriented Programming (OOP*), khususnya penerapan dua kelas, penggunaan *setter* dan *getter*, serta *encapsulation* untuk mengelola data akun secara terstruktur.

Program terdiri dari dua file utama, yaitu *Akun.java* sebagai *class data* dan *tugasAlproPekan7_2511532002.java* sebagai *class utama*.

a) *Akun.java (Class data)*

Class Akun berfungsi sebagai *class data* yang menyimpan variabel, seperti *username, password, email*, dan *PIN*. Seluruh atribut dibuat *private* untuk menjaga keamanan data, kemudian diakses melalui *method setter* dan *getter*.

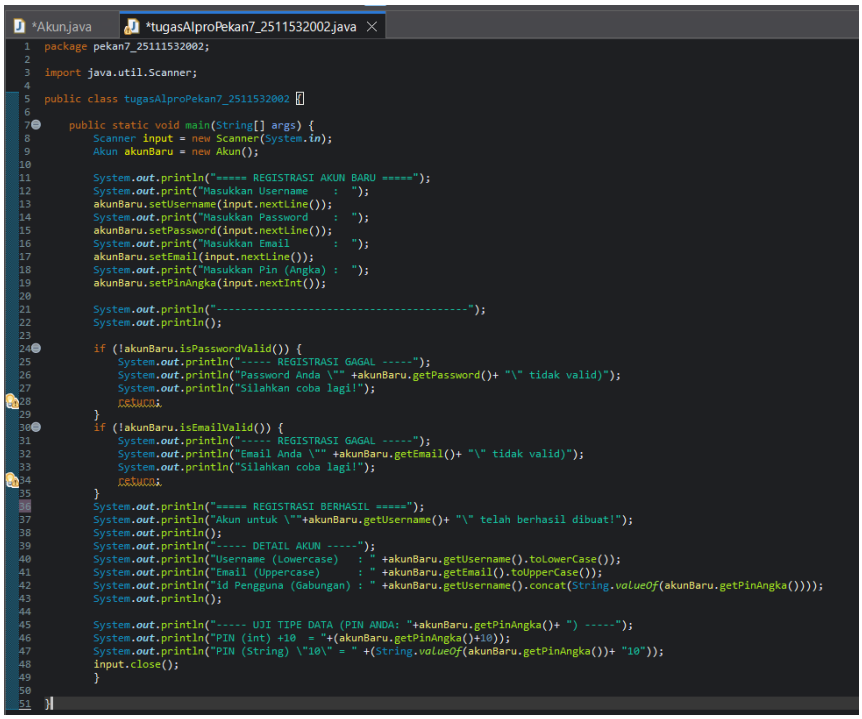
*Akun.java

```java
package pekan7_25111532002;
public class Akun {

    //Variabel Private
    private String username;
    private String password;
    private String email;
    private int pinAngka;

    //Setter
    public void setUsername(String username) {
        this.username = username;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public void setPinAngka(int pinAngka) {
        this.pinAngka = pinAngka;
    }

    //Getter
    public String getUsername() {
        return username;
    }
    public String getPassword() {
        return password;
    }
    public String getEmail() {
        return email;
    }
    public int getPinAngka() {
        return pinAngka;
    }

    public boolean isPasswordValid() {
        return password.length() >= 8;
    }
    public boolean isEmailValid() {
        return email.contains("@") && email.contains(".");
    }
```

2.1. 1 Syntax Program Akun.java (Class Data)

b) tugasAlproPekan7_2511532002.java (Class Utama)

Class Registrasi adalah *class utama* yang mengatur alur dan *output* program. Pada *class* ini, *input* pengguna diambil menggunakan *Scanner*, lalu dikirim ke *object Akun* melalui *setter*. Class ini juga memanggil *method* validasi dari *class Akun* yaitu *isPasswordValid*() dan *isEmailValid*() dan menentukan *output* yang ditampilkan.



```java
package pekan7_25111532002;

import java.util.Scanner;

public class tugasAlproPekan7_2511532002 {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        Akun akunBaru = new Akun();

        System.out.println("===== REGISTRASI AKUN BARU =====");
        System.out.print("Masukkan Username    :  ");
        akunBaru.setUsername(input.nextLine());
        System.out.print("Masukkan Password    :  ");
        akunBaru.setPassword(input.nextLine());
        System.out.print("Masukkan Email       :  ");
        akunBaru.setEmail(input.nextLine());
        System.out.print("Masukkan Pin (Angka) :  ");
        akunBaru.setPinAngka(input.nextInt());

        System.out.println("----------------------------------------");
        System.out.println();

        if (!akunBaru.isPasswordValid()) {
            System.out.println("----- REGISTRASI GAGAL -----");
            System.out.println("Password Anda \"" +akunBaru.getPassword()+ "\" tidak valid)");
            System.out.println("Silahkan coba lagi!");
            return;
        }
        if (!akunBaru.isEmailValid()) {
            System.out.println("----- REGISTRASI GAGAL -----");
            System.out.println("Email Anda \"" +akunBaru.getEmail()+ "\" tidak valid)");
            System.out.println("Silahkan coba lagi!");
            return;
        }
        System.out.println("===== REGISTRASI BERHASIL =====");
        System.out.println("Akun untuk \""+akunBaru.getUsername()+ "\" telah berhasil dibuat!");
        System.out.println();
        System.out.println("----- DETAIL AKUN -----");
        System.out.println("Username (Lowercase)   : " +akunBaru.getUsername().toLowerCase());
        System.out.println("Email (Uppercase)      : " +akunBaru.getEmail().toUpperCase());
        System.out.println("id Pengguna (Gabungan) : " +akunBaru.getUsername().concat(String.valueOf(akunBaru.getPinAngka())));
        System.out.println();

        System.out.println("----- UJI TIPE DATA (PIN ANDA: "+akunBaru.getPinAngka()+ ") -----");
        System.out.println("PIN (int) +10  = "+(akunBaru.getPinAngka()+10));
        System.out.println("PIN (String) \"10\" = " +(String.valueOf(akunBaru.getPinAngka())+ "10"));
        input.close();
        }
}
```

2.1. 2 Syntax Program tugasAlproPekan7_2511532002.java (Class Utama)

2.1.1 Penjelasan Kode

a) Akun.java (Class data)

1. *package pekan7_2511532002;*

   Menjelaskan posisi file berada pada *package* bernama "*pekan7_2511532002*".

2. *public class Akun {*

   Menjelaskan kelas dari *package* pekan pekan7*_2511532002*, yaitu "*Akun*", sebagai *class data.* Untuk menyimpan variabel/ atribut *private* yang dkelola akses nya dengan *setter* dan *getter,* serta *helper methode* yaitu *isPasswordValid()* dan *isEmailValid(.*

3. *Variable Private*

   Variabel private adalah variabel yang hanya bisa diakses dari dalam *class* tempat variabel itu dibuat, dan tidak bisa diakses langsung dari class lain. Fungsi utamanya adalah data lebih aman, tidak bisa sembarangan diubah dari luar. Untuk mengelola atau mengedit data tersebut diperlukan penerapan *methode setter* dan *getter*.

   Berikut merupakan penjelasan untuk tiap-tiap variabel yang digunakan :

```
//Variabel Private
private String username;
private String password;
private String email;
private int pinAngka;
```

2.1. 3 Variabel Private Program Akun.java

   1) *private String username*;

      Variabel *string* ini digunakan untuk menyimpan *username* yang dimasukkan nantinya pada *class utama*.

   2) *private String password*;

      Variabel ini menyimpan *password* akun. *Password* disimpan sebagai *String* agar dapat divalidasi panjangnya dan tidak terbatas pada karakter angka.

   3) private String email;

      Menyimpan alamat *email* pengguna. *Tipe String* digunakan karena *email* merupakan kombinasi huruf, angka, dan simbol.

   4) private int pinAngka;

      Variabel ini menyimpan *PIN* numerik pengguna. PIN bertipe *integrer* karena terdiri dari angka dan digunakan untuk pengujian tipe data di *output*.

4. *Methode Setter*

   Setter adalah *method* yang digunakan untuk mengisi atau mengubah nilai *variabel private* dari luar class.

   Karena variabel *username, password, email*, dan *pinAngka* bersifat *private*, maka perubahan nilainya hanya boleh dilakukan melalui *setter*.

```
//Setter
public void setUsername(String username) {
    this.username = username;
}
```

2.1. 4 Penjeleasan Kode Mthode Setter

   Contoh seperti pada *syntax* di atas yang memberikan nilai baru ke variabel *username* dengan tetap menjaga keamanan data (sesuai prinsip *encapsulation*).

5. *Methode Getter*

   Getter digunakan untuk mengambil/membaca nilai variabel *private*.

```
//Getter
public String getUsername() {
    return username;
}
```

2.1. 5 Penjeleasan Kode Mthode Getter

   Memungkinkan *class* lain (misalnya *class Registrasi*) untuk membaca data tanpa membuka akses langsung ke *variabel*.

6. *Methode* Tambahan

```
public boolean isPasswordValid() {
    return password.length() >= 8;
}
public boolean isEmailValid() {
    return email.contains("@") && email.contains(".");
}
```

2.1. 6 Penjeleasan Kode Methode Tambahan

   Methode berikut digunakan untuk validasi data, membantu mengecek apakah *input* sudah sesuai ketentuan yang dibuat.

b) *tugasAlproPekan7_2511532002* (Class Utama)

1. *Scanner input = new Scanner(System.in);*
   Digunakan untuk membaca *input* dari *user* melalui *keyboard*.
2. *Akun akunBaru = new Akun();*
   Membuat objek dari *class Akun (*class data), yang digunakan untuk menyimpan data *username, password, email*, dan *PIN*.
3. *System.out.print() / System.out.println()*
   Menghasilkan *output* teks. *println* membuat baris baru, sedangkan *print* tetap pada baris yang sama.
4. *akunBaru.setUsername(input.nextLine());*
   Mengambil *input username* dari pengguna dan mengirimkannya ke *class Akun* melalui *method setter*.
5. *akunBaru.setPinAngka(input.nextInt());*
   Mengambil *input* angka (*PIN*), lalu menyimpannya ke objek melalui *setter*.
6. *if (!akunBaru.isPasswordValid()) { ... }*
   Percabangan *if* untuk memeriksa apakah *password valid*. Jika tidak valid, akan menampilkan pesan gagal.
7. *if (!akunBaru.isEmailValid()) { ... }*
   Percabangan untuk memeriksa apakah email valid (dimana harus mengandung @ dan .).
8. *"Akun untuk \"" + akunBaru.getUsername() + "\" telah berhasil dibuat."*
   Mengambil nilai *username* melalui *getter* dan menampilkannya dalam *format output*.
9. *akunBaru.getUsername().toLowerCase()*
   Manipulasi *variabel string* mengubah *username* menjadi huruf kecil/ *lowercase*.
10. *akunBaru.getEmail().toUpperCase()*
    Manipulasi string, mengubah *email* menjadi huruf besar.

11. *akunBaru.getUsername().concat(String.valueOf(akunBaru.getPinAngka()))*
    Menggabungkan *username* + *PIN* untuk membuat *ID* pengguna.
12. *akunBaru.getPinAngka() + 10*
    Operasi aritmetika, menjumlahkan PIN (*tipe int*) dengan *10*.
13. *String.valueOf(akunBaru.getPinAngka()) + "10"*
    Konversi PIN ke String, lalu menggabungkannya dengan teks "*10*".

2.2 Langkah Kerja

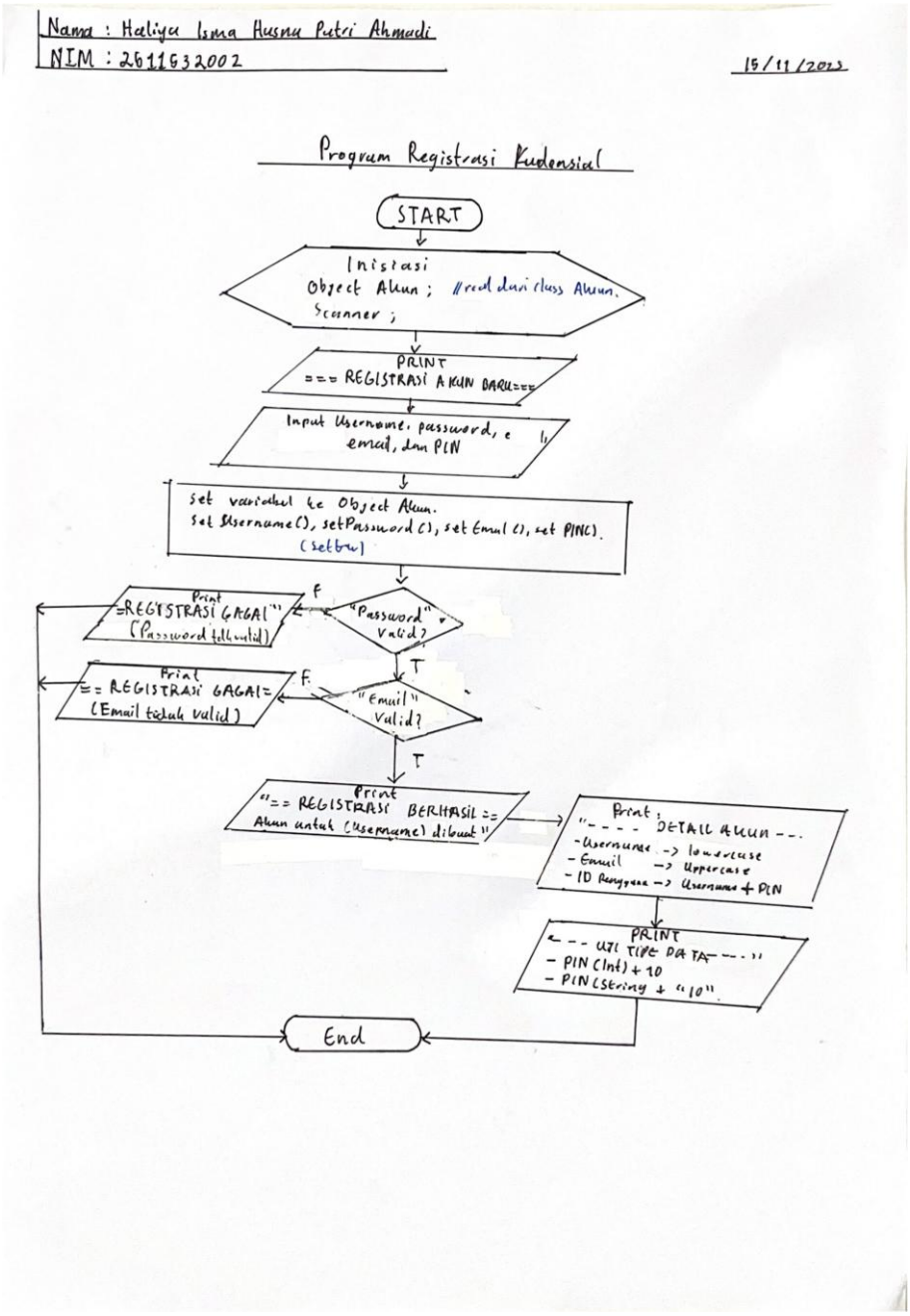
Untuk menyusun sebuah program tentunya perlu untuk merancang algoritma dari program tersebut terlebih dahulu, agar logika program dapat dipahami dengan baik sebelum diubah ke bahasa permrograman. Algoritma tersebut dapat dibuat dalam bentuk bahasa natural terlebih dahulu, lalu *flowchart*, dan *pseudocode* baru selanjutnya diaplikasikan ke bahasa pemrograman.

2.2.1 Bahasa Natural

1. Mulai.
2. Inisiasi variabel: *Scanner*, *Akun*
3. *Print* username, lalu *setUsername()* <- *input*.
4. *print password*, kemudian *setPassword*() <- *input*
5. *Print email*, kemudian *setEmail*()<- input.
6. *Print PIN*, kemudian *setPinAngka()*<- *input*.
7. *RUN* isPasswordValid().
8. *IF* (passwordValid == false) *THEN*:
9. *Print* ("----- REGISTRASI GAGAL -----").
10. *End Program*
11. *RUN isEmailValid*().

12. *IF* (emailValid == false) *THEN*:
13. *Print* ("----- REGISTRASI GAGAL -----").
14. *End Program*
15. *ELSE*
16. *Print* ("===== REGISTRASI BERHASIL =====")
17. *Print ("Akun untuk \" akun "\" telah berhasil dibuat!");*
18. *print* ("----- DETAIL AKUN -----");:
19. *Print Username (toLowerCase())*
20. *Print Email (toUpperCase())*
21. ID = *username +PIN*.
22. *Print (----- UJI TIPE DATA (PIN ANDA : PinAngka) ----)*
23. *Print*: Int.*pinAngka + 10*
24. *Prnnt* : String.Pin + "10"
25. *input.close()*.
26. *Selesai.*

2.2.2 Flowchart



2.1. 7 Flowchart Program Registrasi Kredensial

2.2.3 Pseudocode

Judul

Program Registrasi Kredensial

{ Program digunakan untuk melakukan proses registrasi akun baru dengan validasi password dan email menggunakan dua *class.*}

Deklarasi

Var = *username, password, email* : *String*

Var = *pinAngka : Integer*

Var = *akunBaru : Object (Akun)*

Var = *validPassword, validEmail : Boolean*

Pseudocode

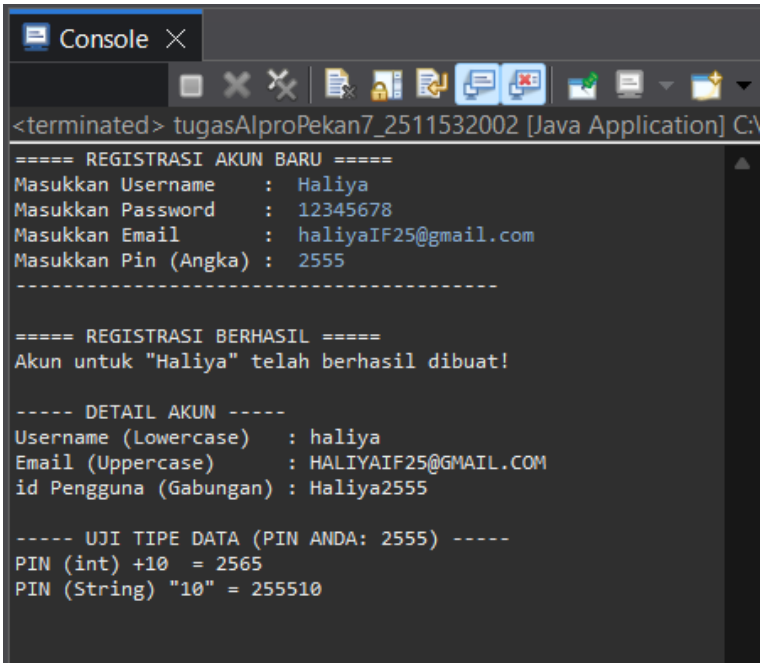
1. *akunBaru <- Akun*
2. *Print "Masukkan Username : "*
3. *akunBaru.setUsername(username) <- Read username*
4. *Print "Masukkan Password : "*
5. *akunBaru.setPassword(password) <- Read password*
6. *Print "Masukkan Email : "*
7. *akunBaru.setEmail(email)<- Read email*
8. *Print "Masukkan PIN (Angka) : "*
9. *akunBaru.setPinAngka(pinAngka)<- Read pinAngka*
10. *validPassword ← akunBaru.isPasswordValid()*
11. *If validPassword == false then*
12. *Print "----- REGISTRASI GAGAL -----"*
13. *Print "Password tidak valid (minimal 8 karakter)."*
14. *End Program*
15. *validEmail ← akunBaru.isEmailValid()*
16. *If validEmail == false then*
17. *Print "----- REGISTRASI GAGAL -----"*
18. *Print "Email tidak valid (harus mengandung @ dan .)"*
19. *end program*

20. *Print "===== REGISTRASI BERHASIL ====="*
21. *Print "Akun untuk" + akunBaru.getUsername() + " telah berhasil dibuat."*
22. *Print "Username (Lowercase): " + akunBaru.getUsername().toLowerCase()*
23. *Print "Email (Uppercase): " + akunBaru.getEmail().toUpperCase()*
24. *Print "ID Pengguna: " + akunBaru.getUsername() concat pinAngka*
25. *Print "PIN (int) + 10 = " + (pinAngka + 10)*
26. *Print "PIN (String) + '10' = " + (String(pinAngka) + "10")*
27. *Close Scanner*
28. *End*

2.1. 8 Pseudocode Program Registrasi Kredensia

2.3 Analisis Hasil

a) Hasil Uji 1 (Kasus Registrasi Berhasil)



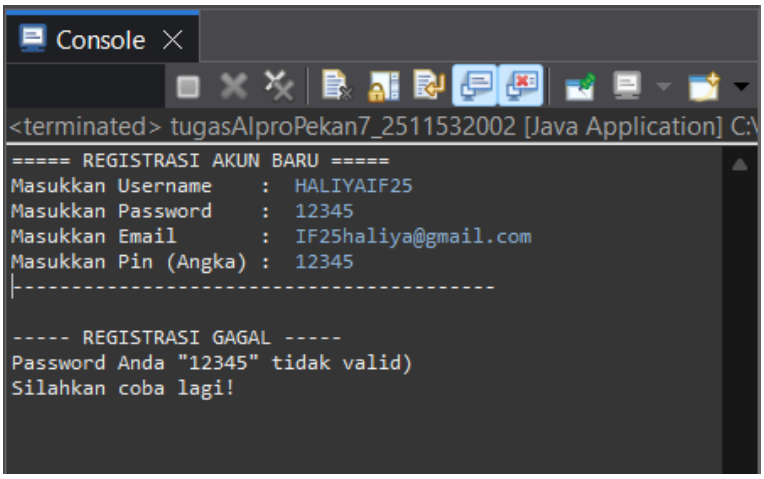
2.1. 9 Hasil Uji 1 (Kasus Registrasi Berhasil)

Pada percobaan pertama, *user* memasukkan:

- *Username: Haliya*
- *Password: 12345678 = ≥ 8 karakter (valid)*
- *Email: haliyaIF25@gmail.com = mengandung karakter '@' dan '.' (valid)*
- *PIN: 2555*

Program menampilkan pesan "REGISTRASI BERHASIL", karena semua syarat terpenuhi. Lalu menampilkan detail akun dimana *Username (lowercase)*, *Email (uppercase)*, *ID* gabungan, dan uji tipe data PIN.

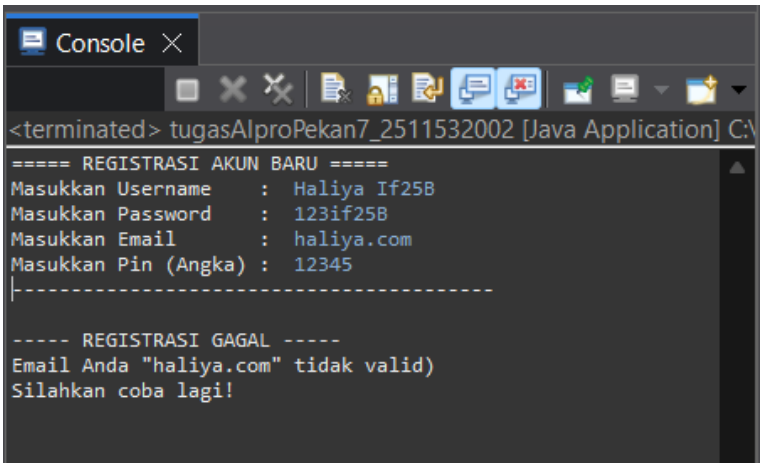
b) Hasil Uji 2 (Kasus Gagal Validasi *Password*)



2.1. 10 Hasil Uji 2 (Kasus Gagal Validasi Password)

Pada percobaan kedua, *password* yang dimasukkan adalah *12345* (kurang dari 8 karakter). Maka, program langsung berhenti dan menampilkan pesan: "REGISTRASI GAGAL, *password* tidak valid". Ini membuktikan bahwa pengecekan *isPasswordValid()* bekerja dengan baik.

c) Hasil Uji 3 (Kasus Gagal Valdiasi *Email*)



2.1. 11 Hasil Uji 3 (Kasus Gagal Valdiasi Email)

Pada percobaan ketiga, *email* yang dimasukkan adalah *haliya.com*, yang tidak mengandung karakter *'@'*. Sehingga program dihentikan dan menampilkan pesan "*REGISTRASI GAGAL, Email tidak valid"*. Menunjukkan bahwa *method isEmailValid*() berjalan sesuai ketentuan, yaitu email harus mengandung '@' dan '.'.

Berdasarkan hasil pengujian, dapat disimpulkan bahwa program berjalan dengan baik dan sesuai ketentuan soal. Validasi yang diterapkan dalam *method isPasswordValid()* dan *isEmailValid()* mampu mendeteksi apakah input pengguna memenuhi aturan. Selain itu *output* yang ditampilkan juga sesuai, menampilkan beberapa bentuk manipulasi *string seperti* program mengubah *username* menjadi huruf kecil menggunakan *method toLowerCase(), email* dalam huruf kapital menggunakan *toUpperCase(), ID user* dengan menggabungkan *username* dan *PIN* menggunakan concat() atau operator +, dan pengujian tipe data *integrer* dan *string.*

BAB III
KESIMPULAN

3.1 Hasil Praktikum

Berdasarkan hasil praktikum dan pengujian program registrasi akun yang telah dilakukan, dapat disimpulkan bahwa program berhasil dijalankan sesuai dengan ketentuan yang diberikan. Program mampu menerima *input*, menyimpan data ke dalam *object* melalui *method setter*, menampilkan data melalui *getter*, serta Sproses validasi menggunakan *method* tambahan.

Program bekerja dengan baik dan sesuai ketentuan dimana:

1. Penggunaan dua kelas yaitu *Akun.java* sebagai *class data* dan *tugasAlproPekan7_2511532002.java* sebagai *class utama*.
2. *Password* dinyatakan valid apabila memiliki panjang minimal 8 karakter.
3. *Email* dinyatakan valid apabila mengandung karakter '@' dan '.'.
4. Manipulasi *string* dengan *toLowerCase(), toUpperCase(),* dan *concat*() atau +.

Secara keseluruhan, program berjalan dengan baik dan mampu menerapkan smua ketentua yang diminta dengan pemanfaatan *methode getter* dan *setter,* dua class, dan validasi data dengan *methode* lainnya.

3.2 Saran

Diharapkan mahasiswa dapat lebih teliti dalam memahami *setter, getter*, serta konsep *encapsulation* agar *output* yang dihasilkan sesuai. Memastikan *class* data dan *class* utama berda dalam *package* yang sama, dan dalam pemanggilan penulisan *class data* tidk salah agar dapat terpanggil. Selain itu, juga perlu memperhatikan logika validasi dan penulisan *syntax* dengan yang tepat. Dengan ketelitian yang lebih baik, hasil program akan lebih akurat, konsisten, dan sesuai dengan ketentuan yang diberikan.

DAFTAR PUSTAKA

[1] W3Schools, “Java Encapsulation,” dokumen daring. Tersedia pada: https://www.w3schools.com/java/java_encapsulation.asp [Diakses: 28 Oktober 2025].

[2] Petanikode, “Java OOP: Mengenal Setter dan Getter,” dokumen daring. Tersedia pada: https://www.petanikode.com/java-oop-setter-getter/ [Diakses: 28 Oktober 2025].

[3] GeeksforGeeks, “Getter and Setter in Java,” dokumen daring. Tersedia pada: https://www.geeksforgeeks.org/java/getter-and-setter-in-java/ [Diakses: 28 Oktober 2025].

[4] W3Schools, “Java private Keyword,” dokumen daring. Tersedia pada: https://www.w3schools.com/java/ref_keyword_private.asp [Diakses: 28 Oktober 2025].

[5] W3Schools, “Java String Methods,” dokumen daring. Tersedia pada: https://www.w3schools.com/java/java_ref_string.asp [Diakses: 28 Oktober 2025].