TUGAS PEMROGRAMAN II

CLI

Oleh:

Rizki Adhitiya Maulana **NIM. 2410817110014**

PROGRAM STUDI TEKNOLOGI INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS LAMBUNG MANGKURAT

NOVEMBER 2025

DAFTAR ISI

DAFTAR ISI ii
DAFTAR GAMBAR iii
DAFTAR TABEL iv
SOAL 1
A. Source Code 1
B. Output Program 14
C. Pembahasan 17

DAFTAR GAMBAR

Gambar 1. Screenshot Proses Pembuatan Karakter 14
Gambar 2. Screenshot Aktivitas 1 14
Gambar 3. Screenshot Aktivitas 2 15
Gambar 4. Screenshot Aktivitas 3 15
Gambar 5. Screenshot Aktivitas 4 16
Gambar 6. Screenshot Status Karakter 16
Gambar 7. Screenshot Keluar dari Program 16

DAFTAR TABEL

Tabel 1. Source Code Player,java 1
Tabel 2. Source Code Activity,java 2
Tabel 3. Source Code Farming,java 3
Tabel 4. Source Code Mining,java 3
Tabel 5. Source Code Fishing,java 4
Tabel 6. Source Code Woodcutting,java 4
Tabel 7. Source Code Tool,java 5
Tabel 8. Source Code WateringCan,java 5
Tabel 9. Source Code Pickaxe,java 6
Tabel 10. Source Code FishingRod,java 6
Tabel 11. Source Code Axe,java 7
Tabel 12. Source Code Usable,java 7
Tabel 13. Source Code MenuHandler,java 8
Tabel 14. Source Code Game,java 11
Tabel 15. Source Code Main.java 13

SOAL

Menggunakan prinsip Abstract class, Interface dan Composition. Buatkan CLI template menggunakan Bahasa Java. Tidak boleh menggunakan Library apapun di luar base Java.

Spec:

1. CLI harus memiliki halaman yang bisa dipilih di menu utama
2. CLI harus memiliki dua fitur memilih menu, scan input user dan menampilkannya.
3. Menggunakan prinsip OOP penyembunyian, main static tidak boleh menjadi GOD class dan harus DUMB
4. Buat report berisikan alasan kalian menggunakan teknik kalian dan insight yang kalian dapatkan

A. Source Code

Tabel 1. Source Code Player.java

```
1  package core;
2
3  public class Player
4  {
5      private final String name;
6      private final String gender;
7      private final String hobby;
8      private int energy;
9
10     public Player(String name, String gender, String
   hobby)
11     {
12         this.name = name;
13         this.gender = gender;
14         this.hobby = hobby;
15         this.energy = 100;
```

```
16      }
17
18      public void reduceEnergy(int amount)
19      {
20          energy -= amount;
21          if (energy < 0)
22          {
23              energy = 0;
24          }
25      }
26
27      public void showStatus()
28      {
29          System.out.println("\n=== STATUS PEMAIN ===");
30          System.out.println("Nama   : " + name);
31          System.out.println("Gender : " + gender);
32          System.out.println("Hobi   : " + hobby);
33          System.out.println("Energi : " + energy);
34      }
35  }
```

Tabel 2. Source Code Activity.java

```
1   package activities;
2
3   import core.Player;
4
5   public abstract class Activity
6   {
7       public abstract void perform(Player player);
8   }
```

Tabel 3. Source Code Farming.java

```
1  package activities;
2
3  import core.Player;
4
5  public class Farming extends Activity
6  {
7      @Override
8      public void perform(Player player)
9      {
10         System.out.println("Kamu menanam tanaman dan
   menyiram tanaman!");
11         player.reduceEnergy(10);
12     }
13 }
```

Tabel 4. Source Code Mining.java

```
1  package activities;
2
3  import core.Player;
4
5  public class Mining extends Activity
6  {
7      @Override
8      public void perform(Player player)
9      {
10         System.out.println("Kamu menambang dan menemukan
   batu!");
11         player.reduceEnergy(15);
12     }
13 }
```

Tabel 5. Source Code Fishing.java

```
1  package activities;
2
3  import core.Player;
4
5  public class Fishing extends Activity
6  {
7      @Override
8      public void perform(Player player)
9      {
10         System.out.println("Kamu memancing dan
   mendapatkan ikan!");
11         player.reduceEnergy(8);
12     }
13 }
```

Tabel 6. Source Code Woodcutting.java

```
1  package activities;
2
3  import core.Player;
4
5  public class Woodcutting extends Activity
6  {
7      @Override
8      public void perform(Player player)
9      {
10         System.out.println("Kamu menebang pohon dan
   mendapatkan kayu!");
11         player.reduceEnergy(12);
12     }
13 }
```

Tabel 7. Source Code Tool,java

```
1  package tools;
2
3  public abstract class Tool
4  {
5      protected String name;
6
7      public Tool(String name)
8      {
9          this.name = name;
10     }
11
12     public String getName()
13     {
14         return name;
15     }
16 }
```

Tabel 8. Source Code WateringCan,java

```
1  package tools;
2
3  public class WateringCan extends Tool implements Usable
4  {
5      public WateringCan()
6      {
7          super("Watering Can");
8      }
9
10     @Override
11     public void use()
12     {
13         System.out.println("Kamu menggunakan " + name +
   " untuk menyiram tanaman...");
```

```
14      }
15  }
```

Tabel 9. Source Code Pickaxe,java

```
1   package tools;
2
3   public class Pickaxe extends Tool implements Usable
4   {
5       public Pickaxe()
6       {
7           super("Pickaxe");
8       }
9
10      @Override
11      public void use()
12      {
13          System.out.println("Kamu menggunakan " + name +
    " untuk memecah batu...");
14      }
15  }
```

Tabel 10. Source Code FishingRod,java

```
1   package tools;
2
3   public class FishingRod extends Tool implements Usable
4   {
5       public FishingRod()
6       {
7           super("Fishing Rod");
8       }
9
10      @Override
11      public void use()
```

```
12      {
13          System.out.println("Kamu menggunakan " + name +
    " untuk memancing ikan...");
14      }
15  }
```

Tabel 11. Source Code Axe,java

```
1   package tools;
2
3   public class Axe extends Tool implements Usable
4   {
5       public Axe()
6       {
7           super("Axe");
8       }
9
10      @Override
11      public void use()
12      {
13          System.out.println("Kamu menggunakan " + name +
    " untuk menebang pohon...");
14      }
15  }
```

Tabel 12. Source Code Usable,java

```
1   package tools;
2
3   public interface Usable
4   {
5       void use();
6   }
```

Tabel 13. Source Code MenuHandler.java

```
1   package core;
2
3   import java.util.Scanner;
4   import activities.*;
5   import tools.*;
6
7   public class MenuHandler
8   {
9       private final Game game;
10      private final Scanner scanner;
11
12      public MenuHandler(Game game, Scanner scanner)
13      {
14          this.game = game;
15          this.scanner = scanner;
16      }
17
18      public void showMainMenu()
19      {
20          while (true)
21          {
22              System.out.println("\n=== STARDUST VALLEY
    CLI ===");
23              System.out.println("1. Lakukan Aktivitas");
24              System.out.println("2. Lihat Status
    Pemain");
25              System.out.println("0. Keluar");
26              System.out.print("Pilih menu: ");
27
28              String choice = scanner.nextLine();
29
30              if (choice.equals("1"))
```

```
31              {
32                  showActivityMenu();
33              }
34              else if (choice.equals("2"))
35              {
36                  game.getPlayer().showStatus();
37              }
38              else if (choice.equals("0"))
39              {
40                  System.out.println("Terima kasih telah
    bermain!");
41                  break;
42              }
43              else
44              {
45                  System.out.println("Pilihan tidak
    valid!");
46              }
47          }
48      }
49
50      private void showActivityMenu()
51      {
52          System.out.println("\n=== PILIH AKTIVITAS
    ===");
53          System.out.println("1. Bertani");
54          System.out.println("2. Menambang");
55          System.out.println("3. Memancing");
56          System.out.println("4. Menebang Pohon");
57          System.out.print("Pilih aktivitas: ");
58
59          String choice = scanner.nextLine();
60
```

```
61          Activity activity = null;
62          Usable tool = null;
63
64          if (choice.equals("1"))
65          {
66              activity = new Farming();
67              tool = new WateringCan();
68          }
69          else if (choice.equals("2"))
70          {
71              activity = new Mining();
72              tool = new Pickaxe();
73          }
74          else if (choice.equals("3"))
75          {
76              activity = new Fishing();
77              tool = new FishingRod();
78          }
79          else if (choice.equals("4"))
80          {
81              activity = new Woodcutting();
82              tool = new Axe();
83          }
84          else
85          {
86              System.out.println("Aktivitas tidak
    dikenal!");
87          }
88
89          if (tool != null)
90          {
91              tool.use();
92          }
```

```
93
94          if (activity != null)
95          {
96              activity.perform(game.getPlayer());
97          }
98      }
99
100 }
```

Tabel 14. Source Code Game,java

```
1   package core;
2
3   import java.util.Scanner;
4
5   public class Game
6   {
7       private final Scanner scanner = new
    Scanner(System.in);
8       private Player player;
9       private MenuHandler menuHandler;
10
11      public void start()
12      {
13          createCharacter();
14          menuHandler = new MenuHandler(this, scanner);
15          menuHandler.showMainMenu();
16      }
17
18      private void createCharacter()
19      {
20          System.out.println("=== SELAMAT DATANG DI
    STARDUST VALLEY ===");
21
```

```
22          System.out.print("Masukkan nama karakter Anda:
    ");
23          String name = scanner.nextLine();
24
25          System.out.print("Pilih gender (L/P): ");
26          String genderInput = scanner.nextLine();
27
28          String gender;
29          if (genderInput.equalsIgnoreCase("L"))
30          {
31              gender = "Laki-laki";
32          }
33          else if (genderInput.equalsIgnoreCase("P"))
34          {
35              gender = "Perempuan";
36          }
37          else
38          {
39              gender = "Tidak diketahui";
40          }
41
42          System.out.print("Masukkan hobi utama karakter
43  Anda: ");
            String hobby = scanner.nextLine();
44
45          this.player = new Player(name, gender, hobby);
46          System.out.println("\nKarakter berhasil
47  dibuat!");
            System.out.println("Nama  : " + name);
48          System.out.println("Gender: " + gender);
49          System.out.println("Hobi  : " + hobby);
50      }
51
```

```
52      public Player getPlayer()
53      {
54          return player;
55      }
56  }
```

Tabel 15. Source Code Main.java

```
1   import core.Game;
2
3   public class Main {
4       public static void main(String[] args) {
5           Game game = new Game();
6           game.start();
7       }
8   }
```

B. Output Program

```
C:\Users\advan\.jdks\openjdk-24.0.2+12-54\bin\java.exe "-javaag
=== SELAMAT DATANG DI STARDUST VALLEY ===
Masukkan nama karakter Anda: Ahdit
Pilih gender (L/P): L
Masukkan hobi utama karakter Anda: Tidor

Karakter berhasil dibuat!
Nama  : Ahdit
Gender: Laki-laki
Hobi  : Tidor
```

Gambar 1. Screenshot Proses Pembuatan Karakter

```
=== STARDUST VALLEY CLI ===
1. Lakukan Aktivitas
2. Lihat Status Pemain
0. Keluar
Pilih menu: 1

=== PILIH AKTIVITAS ===
1. Bertani
2. Menambang
3. Memancing
4. Menebang Pohon
Pilih aktivitas: 1
Kamu menggunakan Watering Can untuk menyiram tanaman...
Kamu menanam tanaman dan menyiram tanaman!
```

Gambar 2. Screenshot Aktivitas 1

```
=== STARDUST VALLEY CLI ===
1. Lakukan Aktivitas
2. Lihat Status Pemain
0. Keluar
Pilih menu: 1

=== PILIH AKTIVITAS ===
1. Bertani
2. Menambang
3. Memancing
4. Menebang Pohon
Pilih aktivitas: 2
Kamu menggunakan Pickaxe untuk memecah batu...
Kamu menambang dan menemukan batu!
```

Gambar 3. Screenshot Aktivitas 2

```
=== STARDUST VALLEY CLI ===
1. Lakukan Aktivitas
2. Lihat Status Pemain
0. Keluar
Pilih menu: 1

=== PILIH AKTIVITAS ===
1. Bertani
2. Menambang
3. Memancing
4. Menebang Pohon
Pilih aktivitas: 3
Kamu menggunakan Fishing Rod untuk memancing ikan...
Kamu memancing dan mendapatkan ikan!
```

Gambar 4. Screenshot Aktivitas 3

```
=== STARDUST VALLEY CLI ===
1. Lakukan Aktivitas
2. Lihat Status Pemain
0. Keluar
Pilih menu: 1

=== PILIH AKTIVITAS ===
1. Bertani
2. Menambang
3. Memancing
4. Menebang Pohon
Pilih aktivitas: 4
Kamu menggunakan Axe untuk menebang pohon...
Kamu menebang pohon dan mendapatkan kayu!
```

Gambar 5. Screenshot Aktivitas 4

```
=== STARDUST VALLEY CLI ===
1. Lakukan Aktivitas
2. Lihat Status Pemain
0. Keluar
Pilih menu: 2

=== STATUS PEMAIN ===
Nama   : Ahdit
Gender : Laki-laki
Hobi   : Tidor
Energi : 55
```

Gambar 6. Screenshot Status Karakter

```
=== STARDUST VALLEY CLI ===
1. Lakukan Aktivitas
2. Lihat Status Pemain
0. Keluar
Pilih menu: 0
Terima kasih telah bermain!
```

Gambar 7. Screenshot Keluar dari Program

C. Pembahasan

- Alur Program

Program dimulai dari kelas "`Main`", yang merupakan titik awal eksekusi. Di kelas ini, objek "`Game`" dibuat, lalu `method start()` dipanggil untuk memulai permainan.

Di dalam kelas "`Game`", program akan meminta pemain untuk membuat karakter baru melalui `method createCharacter()`. Pemain diminta untuk memasukkan nama, gender, dan hobi utama karakter. Semua informasi ini kemudian disimpan dalam objek "`Player`". Selain itu, objek "`MenuHandler`" juga dibuat untuk menangani alur menu di CLI. Kelas "`Game`" memiliki `getter getPlayer()` yang berguna untuk memberikan akses ke objek "`Player`" tanpa harus membuat variabel player menjadi public, sehingga prinsip encapsulation tetap terjaga.

Objek "`Player`" sendiri berisi data dasar karakter seperti nama, gender, hobi, dan energi. `Method showStatus()` digunakan untuk menampilkan status karakter, sedangkan `method reduceEnergy(int amount)` digunakan untuk mengurangi energi pemain saat melakukan aktivitas.

Selanjutnya, "`MenuHandler`" mengatur semua interaksi di CLI. Menu utama menampilkan tiga opsi: melakukan aktivitas, melihat status pemain, atau keluar dari game. Jika pemain memilih melakukan aktivitas, `method showActivityMenu()` dipanggil, dan pemain bisa memilih antara bertani, menambang, memancing, atau menebang pohon. Setiap aktivitas ini otomatis menggunakan alat yang sesuai, misalnya bertani menggunakan WateringCan, menambang menggunakan Pickaxe, memancing menggunakan FishingRod, dan menebang pohon menggunakan Axe. Setelah aktivitas dilakukan, energi pemain berkurang sesuai jenis aktivitas, dan informasi hobi pemain juga ditampilkan.

Aktivitas sendiri diatur melalui kelas "`abstrak Activity`". Setiap jenis aktivitas seperti Farming, Mining, Fishing, dan Woodcutting merupakan subclass yang mengimplementasikan `method perform(Player player)`. Dengan

cara ini, setiap aktivitas dapat memiliki logika khusus sendiri, tapi tetap mengikuti struktur yang sama.

Alat diatur menggunakan kelas "`abstrak Tool`" dan "`interface Usable`". Setiap alat menyimpan nama alat dalam variabel name dan memiliki `method use()` yang diimplementasikan di setiap subclass alat, seperti WateringCan, Pickaxe, FishingRod, dan Axe. "`Interface Usable`" memastikan bahwa semua alat bisa digunakan dengan cara yang konsisten.

Dengan alur ini, kelas "`MenuHandler`" tidak perlu mengetahui detail bagaimana alat atau aktivitas bekerja. Ia hanya memanggil `method use()` pada alat dan `perform()` pada aktivitas, dan semua efeknya akan dijalankan sesuai implementasi masing-masing kelas.

- Alasan Menggunakan Prinsip Ini

  Abstract Class:

  Kelas "`Activity`" dan "`Tool`" dibuat abstrak. Hal ini membuat setiap subclass untuk mengimplementasikan method penting seperti `perform()` dan `use()`. Misalnya, Farming harus mendefinisikan bagaimana aktivitas bertani dilakukan, sedangkan WateringCan harus mendefinisikan bagaimana alat digunakan. Dengan menggunakan abstract class, kita bisa menambahkan aktivitas atau alat baru tanpa harus merombak kode menu atau kelas lain.

  Interface:

  `Interface Usable` diterapkan pada semua alat. `Interface` ini mendeklarasikan `method use()` yang wajib diimplementasikan oleh semua kelas alat. Dengan cara ini, kita menjamin bahwa semua alat memiliki `method use()`, sehingga di "`MenuHandler`" bisa dipanggil secara umum tanpa peduli jenis alatnya. Teknik ini juga menampilkan polimorfisme, karena satu `variabel Usable tool` bisa menampung berbagai jenis alat dan memanggil `method use()` yang berbeda-beda sesuai objeknya.

Composition:

Kelas "Game" memiliki objek "Player" dan "MenuHandler". Kelas "MenuHandler" kemudian menggunakan objek "Player" melalui getter getPlayer() dari "Game". Teknik ini disebut composition, karena kelas yang lebih kompleks terdiri dari objek-objek lain. Dengan cara ini, tanggung jawab antar kelas jelas dan terpisah dimana "Game" bertanggung jawab membuat karakter, "MenuHandler" menangani interaksi pengguna, dan "Player" menyimpan data pemain. Hal ini membuat kode lebih terstruktur, mudah dipelihara, dan memudahkan pengujian unit.

Inheritance:

Konsep inheritance (pewarisan) digunakan agar kelas turunan dapat memanfaatkan kembali kode dari kelas induknya. Misalnya, kelas "Farming", "Mining", "Fishing" dan "Woodcutting" mewarisi sifat dasar dari kelas "Activity", seperti nama aktivitas dan metode umum yang bisa digunakan bersama. Begitu pula "WateringCan", "Pickaxe", "FishingRod" dan "Axe" mewarisi dari kelas "Tool", sehingga semuanya memiliki kerangka kerja yang sama namun bisa menambahkan perilaku unik. Pewarisan ini membantu mengurangi duplikasi kode dan mempermudah pengelolaan fitur baru.

- Insight

Penerapan Prinsip OOP

Penerapan OOP ternyata mampu membuat program lebih terstruktur dan mudah dikembangkan karena setiap kelas punya peran yang jelas dan saling terhubung tanpa tumpang tindih.

Abstraksi yang Meningkatkan Fleksibilitas Program

Penerapan abstract class pada Activity dan Tool membuat sistem lebih mudah dikembangkan tanpa harus mengubah struktur utama. Dengan cara ini, aktivitas baru yang kedepannya dapat ditambahkan seperti Sleep atau Shopping bisa

ditambahkan dengan lancar, menunjukkan bahwa program dirancang untuk terus berkembang dengan mudah.

Interface sebagai Penentu Perilaku Seragam

Melalui `interface Usable`, setiap alat dijamin memiliki fungsi dasar yang sama, yaitu dapat digunakan dengan `method use()`. Pendekatan ini menjaga konsistensi perilaku antar objek sekaligus memberi kebebasan bagi tiap alat untuk memiliki cara kerja yang berbeda sesuai karakteristiknya.

Komposisi yang Menjaga Keteraturan Antar Kelas

Kelas "`Game`" berperan sebagai pusat yang mengatur "`Player`" dan "`MenuHandler`", menunjukkan bahwa sistem dibangun untuk saling melengkapi. Melalui penggunaan getter yang dibuat, setiap kelas dapat berinteraksi dengan aman dan teratur tanpa harus mengakses data internal secara langsung.

Polimorfisme Menciptakan Dinamika Program

Dengan variabel bertipe Activity dan Usable, program dapat menjalankan berbagai perilaku tanpa harus tahu jenis objek yang sebenarnya. Pendekatan ini membuat kode lebih efisien dan logika program terasa lebih alami, karena satu struktur dasar mampu menyesuaikan diri dengan konteks yang berbeda secara elegan.