TUGAS CLI

MATA KULIAH PEMROGRAMAN

Dosen Pengampu: Irham Maulani Abdul Gani S.Kom., M.Kom.

Disusun Oleh:

Sebastian Abe Santoso

2410817210002

PROGRAM STUDI TEKNOLOGI INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS LAMBUNG MANGKURAT

2025

1. 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

2. Kode

Class utama (`Main.java`)

```
 1  package TUGASPEMROCLI;
 2
 3  public class Main {
 4      public static void main(String[] args) {
 5          Gacha gacha = new Gacha();
 6          Process process = new Process(gacha);
 7          GUI gui = new GUI(process);
 8
 9          gui.menuHandler();
10      }
11  }
```

Visual class (`GUI.java`)

```
 1  package TUGASPEMROCLI;
 2
 3  import java.util.List;
 4  import java.util.Scanner;
 5
 6  public class GUI {
 7
 8      private static final Scanner scan = new
    Scanner(System.in);
 9      private final Process process;
10
11      public GUI(Process process) {
12          this.process = process;
13      }
14
15      public void menuHandler() {
16          while (true) {
17              printHeader();
18              printBody();
```

```
            printFooter();

            int choice = scan.nextInt();
            process.handleMenuSelection(choice);
        }
    }

    private void printHeader() {
        System.out.println("\n\n========= HEADER =========");
        System.out.println("Welcome to Genshin Gacha
Simulator");
        System.out.println("You are currently in: " +
process.getCurrentState());
        System.out.println("=========================");
    }

    private void printBody() {
        switch (process.getCurrentState()) {
            case MAIN -> {
                System.out.println("\n\n========== MAIN
==========");
                System.out.println("1. Show Characters");
                System.out.println("2. Show Weapons");
                System.out.println("3. Gacha");
                System.out.println("99. Exit");
             System.out.println("=========================");
            }

            case CHARACTER -> printCharacterInventory();
            case WEAPON -> printWeaponInventory();

            case GACHA -> {
                System.out.println("\n\n========== GACHA
=========");
                System.out.println("1. View Banners");
                System.out.println("2. Roll Gacha");
                System.out.println("3. Banner Details");
                System.out.println("99. Back");
             System.out.println("=========================");
            }
        }
    }

    private void printFooter() {
        System.out.println("\n\n========= FOOTER =========");
        System.out.print("Your choice: ");
    }

    private void printCharacterInventory() {
        System.out.println("\n\n======= CHARACTERS =======");
        List<Character> list =
process.getCharacterInventory();
        if (list.isEmpty()) System.out.println("You have no
characters yet.");
        else list.forEach(c -> System.out.println("- " +
```

```
    c.getName()));
 68         System.out.println("99. Back");
 69     }
 70
 71     private void printWeaponInventory() {
 72         System.out.println("\n\n======== WEAPONS =========");
 73         List<Weapon> list = process.getWeaponInventory();
 74         if (list.isEmpty()) System.out.println("You have no
    weapons yet.");
 75         else list.forEach(w -> System.out.println("- " +
    w.getName()));
 76         System.out.println("99. Back");
 77     }
 78
 79     public static void printSpinMenu() {
 80         System.out.println("\nChoose spin:");
 81         System.out.println("1. Single (1x)");
 82         System.out.println("2. Ten Pull (10x)");
 83         System.out.print("Your choice: ");
 84     }
 85
 86     public static void printBannerMenu() {
 87         System.out.println("\nChoose banner:");
 88         System.out.println("1. Standard");
 89         System.out.println("2. Limited");
 90         System.out.println("3. Weapon");
 91         System.out.print("Your choice: ");
 92     }
 93
 94     public static int getInput() {
 95         return scan.nextInt();
 96     }
 97
 98     public static void printPulledCharacter(Character c) {
 99         System.out.println("\nYou pulled: " + c.getName());
100     }
101
102     public static void
    printPulledTenCharacters(List<Character> list) {
103         System.out.println("\nTen Pull Result:");
104         list.forEach(c -> System.out.println("☆ " +
    c.getName()));
105     }
106 }
```

Control Class (`Process.java`)

```
  1 package TUGASPEMROCLI;
  2
  3 import java.util.ArrayList;
  4 import java.util.List;
  5
  6 public class Process {
  7     public enum MenuState {
  8         MAIN,
```

```
        CHARACTER,
        WEAPON,
        GACHA
    }

    private MenuState currentState = MenuState.MAIN;

    private final Gacha gacha;
    private final List<Character> characterInventory = new ArrayList<>();
    private final List<Weapon> weaponInventory = new ArrayList<>();

    public Process(Gacha gacha) {
        this.gacha = gacha;
    }

    public MenuState getCurrentState() {
        return currentState;
    }

    public void handleMenuSelection(int choice) {
        switch (currentState) {

            case MAIN -> {
                switch (choice) {
                case 1 -> currentState = MenuState.CHARACTER;
                case 2 -> currentState = MenuState.WEAPON;
                case 3 -> currentState = MenuState.GACHA;
                case 99 -> System.exit(0);
                default -> System.out.println("Invalid choice.");
            };
            }

            case CHARACTER, WEAPON, GACHA -> {
                if (choice == 99) {
                    currentState = MenuState.MAIN;
                } else {
                    handleGacha(choice);
                }
            }
        }
    }

    private void handleGacha(int choice) {
        switch (currentState) {
            case GACHA -> {
                switch (choice) {
                    case 1 -> System.out.println("\nAvailable banners:\n1. Standard\n2. Limited\n3. Weapon");

                    case 2 -> {
                        GUI.printSpinMenu();
                        int spin = GUI.getInput();
```

```
                        GUI.printBannerMenu();
                        int bannerChoice = GUI.getInput();
                        if (spin == 1) {
                            Character result =
gachaOnePull(bannerChoice);
                            GUI.printPulledCharacter(result);
                        } else {
                            List<Character> results =
gachaTenPull(bannerChoice);

GUI.printPulledTenCharacters(results);
                        }
                    }
                    case 3 -> System.out.println("\nDrop
rates:\n5★ = 1%\n4★ = 10%\n3★ = 89%");
                }
            }
        }
    }

    public Character gachaOnePull(int bannerChoice) {
        Character result = gacha.roll(bannerChoice);

        if (bannerChoice == 3) {
            weaponInventory.add(result.getWeapon());
            return result;
        }

        characterInventory.add(result);
        weaponInventory.add(result.getWeapon());
        return result;
    }

    public List<Character> gachaTenPull(int bannerChoice) {
        List<Character> results = new ArrayList<>();

        for (int i = 0; i < 10; i++) {
            Character result = gacha.roll(bannerChoice);

            if (bannerChoice == 3) {
                weaponInventory.add(result.getWeapon());
            } else {
                characterInventory.add(result);
                weaponInventory.add(result.getWeapon());
            }

            results.add(result);
        }

        return results;
    }

    public List<Character> getCharacterInventory() {
        return characterInventory;
    }
```

```
111
112     public List<Weapon> getWeaponInventory() {
113         return weaponInventory;
114     }
115 }
```

Class model (`Gacha.java`)

```
  1 package TUGASPEMROCLI;
  2
  3 import java.util.Random;
  4
  5 public class Gacha {
  6     private final Random random = new Random();
  7
  8     private int pullsSinceLastFiveStar = 0;
  9     private boolean guaranteedLimitedNextTime = false;
 10
 11     public Character roll(int bannerChoice) {
 12         return switch (bannerChoice) {
 13             case 1 -> rollStandardBanner();
 14             case 2 -> rollLimitedBanner();
 15             case 3 -> rollWeaponBanner();
 16             default -> null;
 17         };
 18     }
 19
 20     private Character rollStandardBanner() {
 21         pullsSinceLastFiveStar++;
 22
 23         if (pullsSinceLastFiveStar >= 90) {
 24             pullsSinceLastFiveStar = 0;
 25             return fiveStarStandard();
 26         }
 27
 28         int rate = random.nextInt(1000);
 29         if (rate < 10) {
 30             pullsSinceLastFiveStar = 0;
 31             return fiveStarStandard();
 32         }
 33         if (rate < 100) {
 34             return fourStarStandard();
 35         }
 36         return threeStar();
 37     }
 38
 39     private Character fiveStarStandard() {
 40         return new GenericCharacter(
 41                 "Jean (Standard 5★)",
 42                 900, 200,
 43                 new Weapon("Favonius Sword", 200),
 44                 new ElementalSkill("Gale Blade", 350),
 45                 new ElementalBurst("Dandelion Field", 900)
 46         );
 47     }
```

```
48
49      private Character fourStarStandard() {
50          return new GenericCharacter(
51                  "Sucrose (Standard 4★)", 700, 150,
52                  new Weapon("Sacrificial Fragments", 150),
53                  new ElementalSkill("Astable Anemohypostasis
    Creation", 300),
54                  new ElementalBurst("Forbidden Creation", 700)
55          );
56      }
57
58      private Character rollLimitedBanner() {
59          pullsSinceLastFiveStar++;
60
61          if (pullsSinceLastFiveStar >= 90) {
62              pullsSinceLastFiveStar = 0;
63              return fiveStarLimited();
64          }
65
66          int rate = random.nextInt(1000);
67          if (rate < 10) {
68
69              pullsSinceLastFiveStar = 0;
70
71              if (guaranteedLimitedNextTime ||
    random.nextBoolean()) {
72                  guaranteedLimitedNextTime = false;
73                  return fiveStarLimited();
74              } else {
75                  guaranteedLimitedNextTime = true;
76                  return fiveStarStandard();
77              }
78          }
79
80          if (rate < 100) {
81              return fourStarStandard();
82          }
83
84          return threeStar();
85      }
86
87      private Character fiveStarLimited() {
88          return new GenericCharacter("Eula", 1000, 300,
89                  new Weapon("Song of Broken Pines", 300),
90                  new ElementalSkill("Ice Tide Vortex
    (Boosted)", 500),
91                  new ElementalBurst("Glacial Illumination
    (Boosted)", 1500)
92          );
93      }
94
95      private Character rollWeaponBanner() {
96          int rate = random.nextInt(1000);
97
98          if (rate < 10) {
```

```
 99              return new GenericCharacter(
100                      "5★ Weapon Holder",
101                      1_000, 300,
102                      new Weapon("Wolf's Gravestone", 320),
103                      new ElementalSkill("None", 0),
104                      new ElementalBurst("None", 0)
105              );
106          }
107
108          if (rate < 100) {
109              return new GenericCharacter(
110                      "4★ Weapon Holder",
111                      800, 260,
112                      new Weapon("Sacrificial Greatsword",
     180),
113                      new ElementalSkill("None", 0),
114                      new ElementalBurst("None", 0)
115              );
116          }
117
118          return threeStar();
119      }
120
121      private Character threeStar() {
122          return new GenericCharacter(
123                  "3★ Character",
124                  300, 70,
125                  new Weapon("Rusty Sword", 20),
126                  new ElementalSkill("Peck", 30),
127                  new ElementalBurst("Weak Burst", 60)
128          );
129      }
130  }
```

Class model (`Character.java`)

```
 1  package TUGASPEMROCLI;
 2
 3  public abstract class Character implements Damageable {
 4      private String name;
 5      private int hp;
 6      private int atk;
 7      private Weapon weapon;
 8      private ElementalSkill skill;
 9      private ElementalBurst burst;
10
11      public Character(String name, int hp, int atk, Weapon
     weapon, ElementalSkill elementalSkill, ElementalBurst
     elementalBurst) {
12          this.name = name;
13          this.hp = hp;
14          this.atk = atk;
15          this.weapon = weapon;
16          this.skill = elementalSkill;
```

```
17          this.burst = elementalBurst;
18      }
19
20      public String getName() {
21          return name;
22      }
23
24      public int getHp() {
25          return hp;
26      }
27
28      public int getAtk() {
29          return atk;
30      }
31
32      public Weapon getWeapon() {
33          return weapon;
34      }
35
36      public ElementalSkill getSkill() {
37          return skill;
38      }
39
40      public ElementalBurst getBurst() {
41          return burst;
42      }
43
44      @Override
45      public void takeDamage(int damage) {
46          hp -= damage;
47      }
48
49      public String attack() {
50          return name + " attacks and deals " + atk + "
    damage";
51      }
52
53      public abstract String useSkill();
54      public abstract String useBurst();
55  }
```

Class model (`GenericCharacter.java`)

```
1   package TUGASPEMROCLI;
2
3   public class GenericCharacter extends Character {
4       public GenericCharacter(String name, int hp, int atk,
    Weapon weapon,
5                               ElementalSkill elementalSkill,
    ElementalBurst elementalBurst) {
6           super(name, hp, atk, weapon, elementalSkill,
    elementalBurst);
7       }
8
9       @Override
```

```
10      public String useSkill() {
11          return getSkill().activate();
12      }
13
14      @Override
15      public String useBurst() {
16          return getBurst().activate();
17      }
18  }
```

Class model (`ElementalSkill.java`)

```
 1  package TUGASPEMROCLI;
 2
 3  public class ElementalSkill {
 4      private String name;
 5      private int damage;
 6
 7      public ElementalSkill(String name, int damage) {
 8          this.name = name;
 9          this.damage = damage;
10      }
11
12      public String activate() {
            return " used Elemental Skill " + name + " and dealt
    " + damage + " damage!";
13      }
14  }
15
```

Class model (`ElementalBurst.java`)

```
 1  package TUGASPEMROCLI;
 2
 3  public class ElementalBurst {
 4          private String name;
 5          private int damage;
 6
 7          public ElementalBurst(String name, int damage) {
 8              this.name = name;
 9              this.damage = damage;
10          }
11
12          public String activate() {
13              return " used Elemental Burst " + name + " and
    dealt " + damage + " damage!";
14      }
15
16  }
```

Class model (`Weapon.java`)

```
 1  package TUGASPEMROCLI;
 2
```

```
 3  public class Weapon {
 4
 5      private String name;
 6      private int damage;
 7
 8      public Weapon(String name, int damage) {
 9          this.name = name;
10          this.damage = damage;
11      }
12
13      public String getName() {
14          return name;
15      }
16  }
```

Interface (`Damageable.java`)

```
 1  package TUGASPEMROCLI;
 2
 3  public interface Damageable {
 4      void takeDamage(int damage);
 5  }
```

3. Alasan & Insight

Karena saya tidak ingin class Main terlalu kuat, maka saya pisah proses dan `GUI`/*CLI*nya. Sehingga main hanya membuat objek `Gacha`, `Process`, dan `GUI`. Kebetulan saya tertarik dengan metode yang dijelaskan saat pertemuan terakhir, yaitu *Dependency Injection*, jadi saya coba buat injection dari `Main` dengan objek yang telah dibuat. Di sini main dibuat jadi DUMB, hanya membangun `Gacha`, menyuntikkan ke `Process`, lalu menyuntikkan `Process` ke `GUI` melalui constructor. Dengan begitu, `GUI` tidak membuat `Process`, dan `Process` tidak membuat `Gacha`.

Setelah membuat main yang simple, saya lanjut ke `GUI` agar ada fondasi visual sebelum membuat proses. Di sini saya membuat `GUI` untuk *CLI* yang semuanya visual dan tidak berlogik, karena kemaren saya baru dengar tentang *MVC* dari teman saya, karena itu saya berencana memisah *CLI* ini menjadi tiga, visual (`GUI`), controller (`Process`), dan model (`Gacha`, `Character`, `Weapon`, dll).

Dengan struktur ini setiap file berfokus kepada tugasnya sendiri dan tidak peduli bagaimana class lain bekerja. Selain itu, saya juga menggunakan *DI* disini yang baru saya pelajari dan ternyata sangat membantu dalam membuat struktur kode rapi.

Pada saat mulai membangun class `Process`, class model pertama yang saya buat adalah `Gacha`, saat itu class `Process` hanya berisi sekitar 50 baris kode. Saya

mengembangkan `Process` secara bertahap seiring dengan penyelesaian class model lainnya, seperti `Character`, `Weapon`, `GenericCharacter`, `ElementalSkill` dan `ElementalBurst`. Selama proses ini, project ini memberikan saya banyak insight dan pengalaman baru dalam menerapkan konsep OOP .

Awalnya, saya belum mengetahui konsep MVC atau Dependency Injection. Tapi setelah mencoba menerapkannya dalam project ini, saya mulai memahami manfaat keduanya untuk menjaga kode tetap modular. Dengan menerapkan MVC, saya memisahkan menjadi bagian-bagian, yaitu `GUI` menangani seluruh tampilan dan input. `Process` mengelola state dan alur, serta memproses hasil gacha, lalu mengembalikan data untuk ditampilkan `GUI`. Tidak ada pemanggilan print dari dalam `Process`, sehingga peran View dan Controller tegas terpisah.

Yang selanjutnya saya buat setelah `Gacha` adalah `Character`, class ini bersifat abstrak dan mengimplementasi interface `Damageable` yang memberi aturan bahwa karakter dapat terkena damage. Dikarenakan class tersebut abstrak, metode abstrak dan konkret dapat digunakan.

Setelah blueprint `Character` selesai, saya membuat kelas konkret `GenericCharacter` yang dapat diinstansiasi oleh `Gacha` untuk menghasilkan objek karakter. Selanjutnya, saya membuat kelas-kelas model yang merupakan komposisi dari `Character`, yaitu `Weapon`, `ElementalSkill`, dan `ElementalBurst`. Pada desain ini, sebuah karakter memiliki senjata, skill, dan burst. Keputusan memisahkan fitur tersebut ke dalam kelas terpisah dilakukan untuk menghindari *God Class*, menjaga pemisahan tanggung jawab dan memudahkan pengembangan serta perawatan kode, setelah semua model class selesai, saya kembali menyelesaikan class `Process` sepenuhnya.

Insight yang saya dapat secara menyeluruh pada project ini:

1. Main dibuat DUMB
2. Menghindari God Class
3. Model MVC: View (`GUI`) hanya visual + input, tanpa logic, dan controller (`Process`) mengatur alur, tidak menggambar UI
4. Dependency Injection
5. Switch-case multiline block {}
6. enum data type
7. Loose coupling & tight coupling

8. Composition lebih fleksibel dari inheritance: Composition saya gunakan karena `Character` memiliki `Weapon`/`Skill`/`Burst`, bukan menjadi `Weapon`. Dan saya gunakan inheritance karena `GenericCharacter` adalah `Character`.
9. SOLID Principle: S = `GUI` hanya View, `Process` hanya `Controller`, `Gacha` hanya logic gacha, O = Bisa tambah karakter/banner tanpa mengubah class yang sudah ada, L = Semua subclass `Character` bisa dipakai sebagai `Character`, I = `Damageable` hanya punya `takeDamage()`, tidak memaksa method lain, D = `GUI` dan `Process` menerima dependency melalui constructor.