Started on	Tuesday, 16 September 2025, 6:01 AM
State	Finished
Completed on	Thursday, 18 September 2025, 7:36 PM
Time taken	2 days 13 hours
Marks	300.00/300.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Nama File: ColorfulShape.zip

Implementasikan 2 kelas berikut:

Upload jawaban sebagai berkas **ColorfulShape.zip** yang berisi tepat 2 file: `Color.java` dan `Shape.java`.

Catatan: Pastikan setiap output diakhiri oleh endline ("\n") atau menggunakan `println`.

Java 8

ColorfulShape.zip

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	10	Accepted	0.07 sec, 28.92 MB
2	10	Accepted	0.07 sec, 29.04 MB
3	10	Accepted	0.09 sec, 28.95 MB
4	10	Accepted	0.07 sec, 30.33 MB
5	10	Accepted	0.06 sec, 27.92 MB
6	10	Accepted	0.06 sec, 28.05 MB
7	10	Accepted	0.06 sec, 28.38 MB
8	10	Accepted	0.06 sec, 28.95 MB
9	10	Accepted	0.06 sec, 29.12 MB
10	10	Accepted	0.07 sec, 28.79 MB

Question **2**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Sistem Transportasi

Implementasikan program sederhana untuk mengelola kendaraan, supir, dan rute.

Anda akan membuat beberapa kelas Java dan sebuah program utama (`TransportSystem`) untuk mensimulasikan informasi kendaraan, supir, rute, serta interaksi sederhana seperti naik/turun penumpang, mengganti supir, dan estimasi waktu tempuh.

`Driver`

- **Atribut**:
  - `name` (String) — nama supir
  - `licenseNumber` (String) — nomor lisensi
  - `rating` (double) — rating supir (0.0 - 5.0), inisiasi 0.0
- **Method**:
  - `introduce()` — cetak: *Supir: [nama] - Lisensi: [nomor] - Rating: [rating]*
  - `updateRating(double newRating)` — perbarui rating jika valid (0.0–5.0); jika tidak valid cetak pesan error.

`Route`

- **Atribut**:
  - `startPoint` (String)
  - `destination` (String)
  - `distance` (int) — km
  - `averageSpeed` (int) — km/jam (untuk estimasi waktu)
- **Method**:
  - `showRoute()` — cetak: *[start] >> [destination] ([distance] km)* dan estimasi waktu
  - `estimateTravelTime()` — kembalikan waktu (jam) = distance / averageSpeed; jika `averageSpeed <= 0` kembalikan nilai invalid (-1)

`Vehicle`

- **Atribut**:
  - `plateNumber` (String)
  - `type` (String) — contoh: Bus, Taksi
  - `capacity` (int) — kapasitas maksimal penumpang
  - `currentLoad` (int) — jumlah penumpang sekarang (inisialisasi 0)
  - `driver` (Driver) — objek supir
  - `route` (Route) — objek rute
- **Method**:
  - `showInfo()` — cetak informasi kendaraan, supir, rute, estimasi waktu, dan `currentLoad/capacity`
  - `changeDriver(Driver newDriver)` — ganti supir
  - `assignRoute(Route newRoute)` — ganti rute
  - `boardPassenger(int count)` — jika kapasitas cukup tambahkan `currentLoad`, jika tidak cetak pesan gagal
  - `alightPassenger(int count)` — jika `count <= currentLoad` kurangi, jika tidak cetak pesan gagal

`TransportSystem`

- **Atribut**:
  - `name` (String)
  - `vehicles` (List of String)
- **Method**:
  - `addVehicle(Vehicle vehicle)` — Menambahkan kendaraan ke sistem
  - `findVehicleByDriver(String driverName)` — Mencari kendaraan berdasarkan nama supir
  - `totalPassengerCapacity()` — mendapatkan total kapasitas penumpang di sistem
  - `showAllVehicles()` — Menampilkan seluruh data kendaraan di sistem

Jangan lupa endline di akhir statement.

Kumpulkan kelas kelas tersebut dalam **transportsystem.zip**

Java 8

transportsystem.zip

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	10	Accepted	0.06 sec, 28.88 MB
2	10	Accepted	0.07 sec, 29.93 MB
3	10	Accepted	0.06 sec, 28.91 MB
4	10	Accepted	0.06 sec, 27.99 MB
5	10	Accepted	0.07 sec, 26.20 MB
6	10	Accepted	0.07 sec, 28.92 MB
7	10	Accepted	0.07 sec, 28.79 MB
8	10	Accepted	0.07 sec, 28.82 MB
9	10	Accepted	0.06 sec, 27.96 MB
10	10	Accepted	0.06 sec, 28.93 MB

Question **3**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Nama File: Developer.java, Avatar.java, Game.java, Player.java

Implementasikan 4 kelas berikut: Avatar.java, Developer.java, Game.java, dan Player.java.

Upload jawaban sebagai berkas **roblox.zip** yang berisi tepat 4 file: `Developer.java` , `Avatar.java`, `Game.java` , `Player.java`.

Catatan: Pastikan setiap output diakhiri oleh endline ("\n") atau menggunakan `println`. File - file di atas akan dicompile dalam satu package yang sama, apabila diperlukan, silahkan ubah public-private identifier dari setiap attribut dan method yang ada di dalam kelas - kelas.

Java 8

roblox.zip

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	10	Accepted	0.07 sec, 28.34 MB
2	10	Accepted	0.07 sec, 28.02 MB
3	10	Accepted	0.06 sec, 28.89 MB
4	10	Accepted	0.07 sec, 30.41 MB
5	10	Accepted	0.06 sec, 28.18 MB
6	10	Accepted	0.06 sec, 28.32 MB
7	10	Accepted	0.07 sec, 26.28 MB
8	10	Accepted	0.07 sec, 28.89 MB
9	10	Accepted	0.07 sec, 29.54 MB
10	10	Accepted	0.07 sec, 28.58 MB

◄ Pasca Praktikum 1

Jump to...

Pra Praktikum 2 OOP Semester Ganjil 2025/2026

Color.java

```
public class Color {
    private int red;
    private int green;
    private int blue;

    /**
     * Konstruktor Color
     *
     * @param r nilai red (0-255)
     * @param g nilai green (0-255)
     * @param b nilai blue (0-255)
     */
    public Color(int r, int g, int b) {
        // TODO: Print constructor message with format
        // "Color is being built with RGB(r, g, b)"
        // Example: if r=255, g=0, b=0, print "Color is being
built with RGB(255, 0, 0)"

        // TODO: Initialize the red, green, and blue instance
variables with the
        // parameter values
    }

    @Override
    /**
     * toString
     *
     * fungsi ini berguna untuk mengembalikan representasi
string dari objek Color
     * usage:
     * System.out.println(colorObject);
     *
     * pada umumnya, ketika sebuah objek dicetak menggunakan
System.out.println atau
     * metode lainnya, metode toString() akan dipanggil secara
otomatis untuk
     * mendapatkan representasi string dari objek tersebut.
     *
     * @return string representasi warna dalam format
"RGB(red, green, blue)"
     */
```

```
    public String toString() {
        // TODO: Return a string in the format "RGB(red,
green, blue)"
        // Example: if red=255, green=128, blue=64, return
"RGB(255, 128, 64)"
        return null;
    }

    /**
     * getRed
     *
     * @return nilai red
     */
    public int getRed() {
        // TODO: Return the red component value
    }

    /**
     * getGreen
     *
     * @return nilai green
     */
    public int getGreen() {
        // TODO: Return the green component value
    }

    /**
     * getBlue
     *
     * @return nilai blue
     */
    public int getBlue() {
        // TODO: Return the blue component value
    }

    /**
     * setColor
     * Mengubah nilai warna
     *
     * @param r nilai red (0-255)
     * @param g nilai green (0-255)
     * @param b nilai blue (0-255)
     */
    public void setColor(int r, int g, int b) {
```

```
        // TODO: Set the red, green, and blue instance
variables to the parameter values
    }
}
```

Shape.java

```
public class Shape {
    private Color color;
    private String name;

    /**
     * Konstruktor Shape
     *
     * @param color
     * @param name
     */
    public Shape(Color color, String name) {
        // TODO: Print constructor message with format
        // "Shape is being built with color [color] and name
'[name]'"
        // Example: if color is RGB(255, 0, 0) and name is
"Circle", print
        // "Shape is being built with color RGB(255, 0, 0) and
name 'Circle'"

        // TODO: Initialize the color and name instance
variables
        // with the parameter values
    }

    /**
     * Default constructor for Shape
     */
    public Shape() {
        // TODO: Print constructor message
        // "Shape is being built with default color and name
'Default'"

        // TODO: Initialize color with a new Color(0, 0, 0)
        // TODO: Initialize name with "Default"
    }

    /**
     * Constructor with name parameter
```

```
     *
     * @param name the name of the shape
     */
    public Shape(String name) {
        // TODO: Print constructor message
        // "Shape is being built with default color and name
'[name]'"
        // Example: if name is "Circle", print
        // "Shape is being built with default color and name
'Circle'"

        // TODO: Initialize color with a new Color(0, 0, 0)
        // TODO: Initialize name with the parameter value
    }

    /**
     * Constructor with RGB parameters
     *
     * @param r red component (0-255)
     * @param g green component (0-255)
     * @param b blue component (0-255)
     */
    public Shape(int r, int g, int b) {
        // TODO: Print constructor message
        // "Shape is being built with color RGB(r, g, b) and
default name 'Default'"
        // Example: if r=255, g=0, b=0, print
        // "Shape is being built with color RGB(255, 0, 0) and
default name 'Default'"

        // TODO: Initialize color with a new Color(r, g, b)
        // TODO: Initialize name with "Default"
    }

    /**
     * getColor
     *
     * @return color
     */
    public Color getColor() {
        // TODO: Return the color instance variable
    }

    /**
```

```
     * setColor
     *
     * @param color
     */
    public void setColor(Color color) {
        // TODO: Set the color instance variable to the
parameter value
    }

    /**
     * getName
     *
     * @return name
     */
    public String getName() {
        // TODO: Return the name instance variable
    }

    /**
     * setName
     *
     * @param name
     */
    public void setName(String name) {
        // TODO: Set the name instance variable to the
parameter value
    }
}
```

Driver.java

```
public class Driver {
    private String name;
    private String licenseNumber;
    private double rating; // rating 0.0 - 5.0

    public Driver(String name, String licenseNumber) {
     /**
      * TODO: Buatlah konstruktor untuk driver
      *   sesuai dengan attribut yang dimiliki oleh kelas
driver,
      *   Rating diset default ke nol.
      * */
    }
```

```
    public void introduce() {
     /**
      * TODO: Fungsi ini untuk memperkenalkan driver dengan
format
      *   "Supir: {name} - Lisensi: {license} - Rating:
{rating}"
      * */
    }

    public void updateRating(double newRating) {
     /**
      * TODO: Fungsi ini untuk melakukan update pada rating
      *   Jika gagal, output saja "Rating harus antara 0
sampai 5"
      * */
    }

    public String getName() {
        return name;
    }

    public String getLicenseNumber() {
        return licenseNumber;
    }

    public double getRating() {
        return rating;
    }
}
```

Route.java

```
public class Route {
    private String startPoint;
    private String destination;
    private int distance;
    private int averageSpeed;

    public Route(String startPoint, String destination, int
distance, int averageSpeed) {
     /**
      * TODO: Buatlah konstruktor untuk kelas route ini
      * */
    }
```

```
    public void showRoute() {
         /**
          * TODO: Menampilkan informasi rute dengan format,
          *         "{startPoint} >> {destination} ({distance}
km)
          *         Estimasi waktu tempuh: {time} jam"
          *         Waktu tempuh ditulis sampai 1 angka di
belakang koma.
          * */
    }

    public double estimateTravelTime() {
         /**
          * TODO: Fungsi ini memprediksi berapa lama rute
ditempuh dengan
          * kecepatan rata rata
          * */
    }

    public String getStartPoint() {
        return startPoint;
    }

    public String getDestination() {
        return destination;
    }

    public int getDistance() {
        return distance;
    }

    public int getAverageSpeed() {
        return averageSpeed;
    }
}
```

Vehicle.java

```
public class Vehicle {
    private String plateNumber;
    private String type;
    private int capacity;
    private int currentLoad;
    private Driver driver;
    private Route route;
```

```
    public Vehicle(String plateNumber, String type, int
capacity, Driver driver, Route route) {
        /**
      * TODO: Buatlah konstruktor untuk kelas vehicle dengan
currentLoad adalah 0
      * */
    }

    public void showInfo() {
     /**
      * TODO: Tampilkan info dari vehicle
      *        "Kendaraan: Avanza (B 1212 FUV)
      *        Supir: Lina - Lisensi: ALD1122 - Rating: 3.0
      *        Rute: Jakarta >> Tangerang (30 km)
      *        Estimasi waktu tempuh: 0.6 jam
      *        Penumpang: 0/7"
      * */
    }

    public void changeDriver(Driver newDriver) {
     /**
      * TODO: Melakukan perubahan driver di kendaraan
      * */
    }

    public void assignRoute(Route newRoute) {
          /**
      * TODO: Menerapkan rute baru pada  kendaraan
      * */
    }

    public void boardPassenger(int count) {
          /**
      * TODO: Buatlah fungsi yang mesimulasikan penumpang yang
naik ke kendaraan,
      *   fungsi ini akan mengubah attribut current load.
      *   Pesan gagal "Gagal: kapasitas kendaraan tidak
cukup!"
      * */
    }

    public void alightPassenger(int count) {
          /**
```

```
      * TODO: Buatlah fungsi yang mesimulasikan penumpang yang
turun dari kendaraan,
      *   fungsi ini akan mengubah attribut current load.
      *   Pesan gagal "Gagal: jumlah penumpang turun melebihi
yang ada!"
      * */
    }

    public Driver getDriver(){
     return this.driver;
    }

    public int getCapacity(){
     return this.capacity;
    }
    public void setDriver(Driver driver){
     this.driver = driver;
    }
}
```

TransportSystem.java

```
import java.util.ArrayList;

public class TransportSystem {
    private String name;
    private ArrayList<Vehicle> vehicles;

    public TransportSystem(String name) {
     /**
      * TODO: Buatlah konstruktor untuk sistem transportasi
      * */
    }

    public void addVehicle(Vehicle vehicle) {
          /**
      * TODO: Fungsinya untuk menambahkan objek kendaraan ke
list kendaraan
      * */
    }

    public Vehicle findVehicleByDriver(String driverName) {
     /**
      * TODO: Fungsi untuk mencari kendaraan berdasarkan nama
dari driver,
```

```
      * kembalikan nulljika tidak ditemukan
      * */
    }

    public int totalPassengerCapacity() {
     /**
      * TODO: Menghitung seluruh kapasitas yang tersedia
      * */
    }

    public void showAllVehicles() {
        /**
      * TODO: Menampilkan seluruh info kendaraan dengan format
      *   "=== Transport System: {name} ===
      *   {info vehicles}
      *   --------------------
      *   {info vehicles}
      *   --------------------
      *   ...
      *   "
      *   Baris terakhir disertai garis juga yah.
      *   jika kendaraan kosong, keluarkan "No vehicles in the
system
      * */
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public ArrayList<Vehicle> getVehicles() {
        return vehicles;
    }
}
```

Avatar.java

```
class Avatar {
    // TODO: Deklarasi atribut: namaSkin (String), level (int)

    // TODO: Constructor set namaSkin dan level
```

```
    Avatar(String namaSkin, int level) {
        // HINT: this.namaSkin = namaSkin; dst.
    }

    // TODO: Naikkan level (level++) dan cetak:
    // "Skin <namaSkin> naik ke level <level>."
    void upgradeLevel() {
        // HINT: Urutan: tingkatkan level, lalu cetak
    }
}
```

Developer.java

```
class Developer {
    // TODO: Deklarasikan atribut: namaDev (String), rating
(double)

    // TODO: Buat constructor untuk menginisialisasi namaDev
dan rating
    Developer(String namaDev, double rating) {
        // HINT: this.namaDev = namaDev; dst.
    }

    // TODO: Cetak informasi developer sesuai format:
    // "Developer: <namaDev> | Rating: <rating>."
    void infoDev() {
        // HINT: Gunakan System.out.println(...)
    }
}
```

Game.java

```
class Game {
    // TODO: Deklarasi atribut: namaGame (String), genre
(String), developer
    // (Developer)
    // TODO: Deklarasi atribut: playerCount (int) awal 0
    // TODO: Deklarasi atribut statik: totalGame (int)

    // TODO: Constructor set semua field dan increment
totalGame
    Game(String namaGame, String genre, Developer developer) {
        // HINT: this.namaGame = ...; dst.
    }
```

```
    // TODO: Saat player join, increment playerCount dan
cetak:
    // "<username> bergabung ke game <namaGame>."
    void joinGame(Player p) {
        // HINT: Akses p.username
    }

    // TODO: Kembalikan totalGame
    static int getTotalGame() {
        return 0; // ganti dengan nilai dari totalGame
    }
}
```

Player.java

```
class Player {
    // TODO: Deklarasikan atribut: username (String), avatar
(Avatar), game (Game)
    // TODO: Deklarasikan static int totalPlayer untuk
menghitung total player
    // dibuat

    // TODO: Constructor set username dan avatar, dan
increment totalPlayer
    Player(String username, Avatar avatar) {
        // HINT: this.username = username; dst.
    }

    // TODO: Method joinGame: set game ke g kemudian panggil
g.joinGame(this)
    void joinGame(Game g) {
    }

    // TODO: Tampilkan profil sesuai format:
    // Username: <username>.
    // Avatar: <namaSkin> (Lv.<level>).
    // Sedang bermain: <namaGame>.
    // ATAU jika belum gabung: Belum bergabung ke game.
    // Diakhiri 1 baris kosong
    void showProfile() {
        // HINT: Cek game == null
    }

    // TODO: Kembalikan totalPlayer
    static int getTotalPlayer() {
```

```
        return 0; // ganti dengan nilai dari totalPlayer
    }
}
```