UNIVERSITAS PESANTREN TINGGI DARUL ULUM

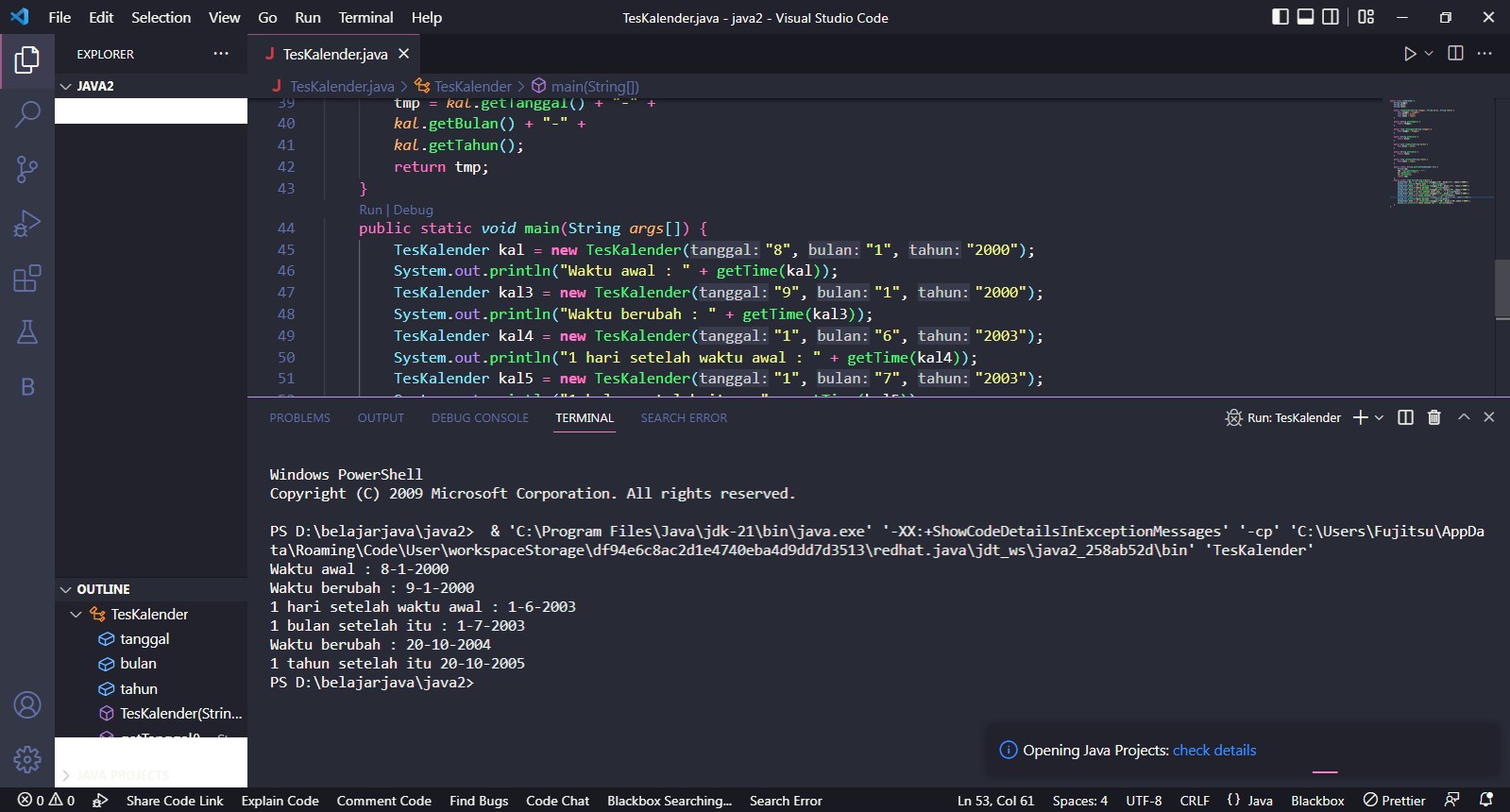
Nama :AHMAD REZA ASARI

Nim :4123019

Kelas : A

Semester : 2

LATIHAN



public class TesKalender {

    String tanggal;

    String bulan;

    String tahun;

    public TesKalender(String *tanggal*, String *bulan*, String *tahun*) {

*this*.tanggal = *tanggal*;

*this*.bulan = *bulan*;

*this*.tahun = *tahun*;

    }

    public String getTanggal() {

        return tanggal;

    }

    public *void* setTanggal(String *tanggal*) {

*this*.tanggal = *tanggal*;

    }

    public String getBulan() {

        return bulan;

    }

    public *void* setBulan(String *bulan*) {

*this*.bulan = *bulan*;

    }

    public String getTahun() {

        return tahun;

    }

    public *void* setTahun(String *tahun*) {

*this*.tahun = *tahun*;

    }

    public static String getTime(TesKalender *kal*) {

        String tmp;

        tmp = *kal*.getTanggal() + "-" +

*kal*.getBulan() + "-" +

*kal*.getTahun();

        return tmp;

    }

    public static *void* main(String *args*[]) {

        TesKalender kal = **new** TesKalender("8", "1", "2000");

        System.out.println("Waktu awal : " + getTime(kal));

        TesKalender kal3 = **new** TesKalender("9", "1", "2000");

        System.out.println("Waktu berubah : " + getTime(kal3));

        TesKalender kal4 = **new** TesKalender("1", "6", "2003");

        System.out.println("1 hari setelah waktu awal : " + getTime(kal4));

        TesKalender kal5 = **new** TesKalender("1", "7", "2003");

        System.out.println("1 bulan setelah itu : " + getTime(kal5));

        TesKalender kal6 = **new** TesKalender("20", "10", "2004");

        System.out.println("Waktu berubah : " + getTime(kal6));

        TesKalender kal2 = **new** TesKalender("20", "10","2005");

        System.out.println("1 tahun setelah itu " + getTime(kal2));

    }

}

1. **Deklarasi Kelas TesKalender**:

javaCopy code

public class TesKalender {

Program dimulai dengan deklarasi kelas **TesKalender**.

1. **Variabel Instance**:

javaCopy code

String tanggal; String bulan; String tahun;

Tiga variabel instance (**tanggal**, **bulan**, **tahun**) digunakan untuk menyimpan informasi tentang tanggal.

1. **Konstruktor**:

javaCopy code

public TesKalender(String tanggal, String bulan, String tahun) { this.tanggal = tanggal; this.bulan = bulan; this.tahun = tahun; }

Konstruktor digunakan untuk membuat objek **TesKalender** baru dengan nilai awal yang diberikan.

1. **Metode Getter dan Setter**:

javaCopy code

public String getTanggal() { return tanggal; } public void setTanggal(String tanggal) { this.tanggal = tanggal; } public String getBulan() { return bulan; } public void setBulan(String bulan) { this.bulan = bulan; } public String getTahun() { return tahun; } public void setTahun(String tahun) { this.tahun = tahun; }

Metode-metode getter digunakan untuk mengambil nilai dari variabel instance, sementara metode-metode setter digunakan untuk mengatur nilai variabel instance.

1. **Metode getTime**:

javaCopy code

public static String getTime(TesKalender kal) { String tmp; tmp = kal.getTanggal() + "-" + kal.getBulan() + "-" + kal.getTahun(); return tmp; }

Metode **getTime** adalah metode statis yang mengambil objek **TesKalender** sebagai argumen dan mengembalikan string yang mewakili tanggal dalam format "tanggal-bulan-tahun".

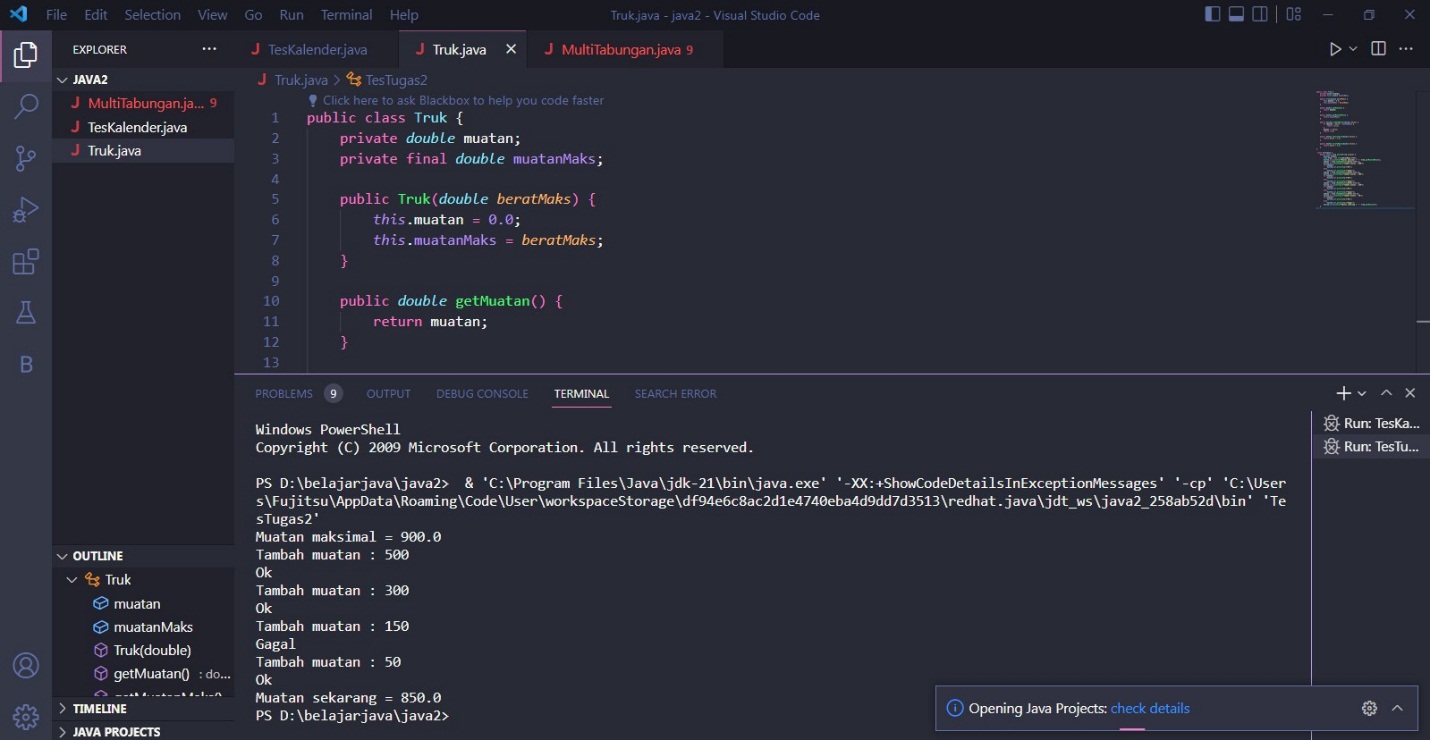
1. **Metode main**:

javaCopy code

public static void main(String args[]) { TesKalender kal = new TesKalender("8", "1", "2000"); System.out.println("Waktu awal : " + getTime(kal)); // Beberapa contoh operasi pada tanggal // ... }

Metode **main** membuat objek **TesKalender** dengan tanggal awal, dan kemudian mencetak tanggal tersebut menggunakan metode **getTime**. Selanjutnya, beberapa operasi sederhana seperti perubahan tanggal, bulan, dan tahun dilakukan dan hasilnya dicetak.

TUGAS

1. 

public class Truk {

    private *double* muatan;

    private final *double* muatanMaks;

    public Truk(*double* *beratMaks*) {

*this*.muatan = 0.0;

*this*.muatanMaks = *beratMaks*;

    }

    public *double* getMuatan() {

        return muatan;

    }

    public *double* getMuatanMaks() {

        return muatanMaks;

    }

    public *boolean* tambahMuatan(*double* *berat*) {

        if (muatan + *berat* > muatanMaks) {

            return false;

        }

        muatan += *berat*;

        return true;

    }

    public *double* newtsToKilo(*double* *berat*) {

        return *berat* / 9.8;

    }

    public *double* kiloToNewts(*double* *berat*) {

        return *berat* \* 9.8;

    }

}

 class TesTugas2 {

    public static *void* main(String *args*[]) {

*boolean* status;

        Truk truk = **new** Truk(900);

        System.out.println("Muatan maksimal = " + truk.getMuatanMaks());

        status = truk.tambahMuatan(500.0);

        System.out.println("Tambah muatan : 500");

        if (status)

            System.out.println("Ok");

        else

            System.out.println("Gagal");

        status = truk.tambahMuatan(300.0);

        System.out.println("Tambah muatan : 300");

        if (status)

            System.out.println("Ok");

        else

            System.out.println("Gagal");

        status = truk.tambahMuatan(150.0);

        System.out.println("Tambah muatan : 150");

        if (status)

            System.out.println("Ok");

        else

            System.out.println("Gagal");

        status = truk.tambahMuatan(50.0);

        System.out.println("Tambah muatan : 50");

        if (status)

            System.out.println("Ok");

        else

            System.out.println("Gagal");

        System.out.println("Muatan sekarang = " + truk.getMuatan());

    }

}

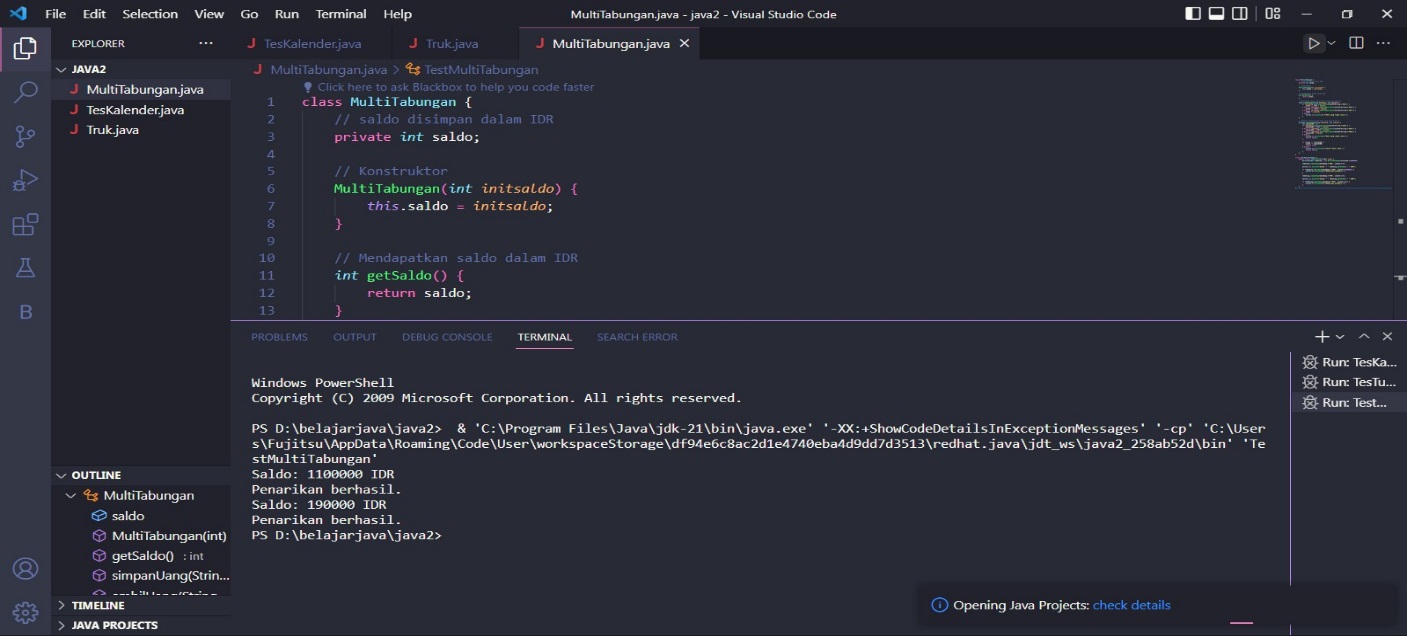
**Kelas Truk:**

1. **Variabel Instance**:
   * **muatan**: Variabel yang menyimpan berat muatan truk.
   * **muatanMaks**: Variabel final yang menyimpan berat maksimum yang dapat ditampung oleh truk.
2. **Konstruktor**:
   * Menerima parameter **beratMaks** yang merupakan berat maksimum yang dapat ditampung oleh truk. Pada konstruktor, **muatan** diinisialisasi dengan nilai **0.0**, dan **muatanMaks** diinisialisasi dengan nilai dari parameter **beratMaks**.
3. **Metode getMuatan**:
   * Mengembalikan nilai dari variabel **muatan**.
4. **Metode getMuatanMaks**:
   * Mengembalikan nilai dari variabel **muatanMaks**.
5. **Metode tambahMuatan**:
   * Menerima parameter **berat** yang merupakan berat muatan yang akan ditambahkan ke truk.
   * Memeriksa apakah penambahan muatan akan melebihi berat maksimum yang diizinkan (**muatan + berat > muatanMaks**).
   * Jika tidak melebihi, maka muatan ditambahkan dan metode mengembalikan **true**, jika melebihi, maka muatan tidak ditambahkan dan metode mengembalikan **false**.
6. **Metode newtsToKilo**:
   * Menerima parameter **berat** dalam satuan Newton dan mengonversinya menjadi kilogram.
7. **Metode kiloToNewts**:
   * Menerima parameter **berat** dalam satuan kilogram dan mengonversinya menjadi Newton.

**Kelas TesTugas2:**

1. **Metode main**:
   * Membuat objek **truk** dari kelas **Truk** dengan berat maksimum 900.
   * Menampilkan berat maksimum truk menggunakan metode **getMuatanMaks**.
   * Menambahkan muatan ke truk menggunakan metode **tambahMuatan**, dan menampilkan pesan apakah penambahan muatan berhasil atau gagal.
   * Menampilkan berat muatan sekarang menggunakan metode **getMuatan**.

Program ini mengilustrasikan bagaimana sebuah kelas dapat digunakan untuk merepresentasikan objek dan melakukan operasi pada objek tersebut, seperti menambahkan muatan ke truk dan menampilkan berat muatan.



Top of Form

class MultiTabungan {

    // saldo disimpan dalam IDR

    private *int* saldo;

    // Konstruktor

    MultiTabungan(*int* *initsaldo*) {

*this*.saldo = *initsaldo*;

    }

    // Mendapatkan saldo dalam IDR

*int* getSaldo() {

        return saldo;

    }

    // Menyimpan jumlah dalam mata uang yang dipilih

*void* simpanUang(String *mataUang*, *int* *jumlah*) {

        if (*mataUang*.equalsIgnoreCase("USD")) {

            saldo += 9000 \* *jumlah*;

        } else if (*mataUang*.equalsIgnoreCase("AUD")) {

            saldo += 10000 \* *jumlah*;

        } else if (*mataUang*.equalsIgnoreCase("IDR")) {

            saldo += *jumlah*;

        } else {

            System.out.println("Mata uang tidak valid.");

        }

    }

    // Menarik jumlah dalam mata uang yang dipilih

*boolean* ambilUang(String *mataUang*, *int* *jumlah*) {

*int* jumlahIDR = 0;

        if (*mataUang*.equalsIgnoreCase("USD")) {

            jumlahIDR = 9000 \* *jumlah*;

        } else if (*mataUang*.equalsIgnoreCase("AUD")) {

            jumlahIDR = 10000 \* *jumlah*;

        } else if (*mataUang*.equalsIgnoreCase("IDR")) {

            jumlahIDR = *jumlah*;

        } else {

            System.out.println("Mata uang tidak valid.");

            return false;

        }

        if (saldo >= jumlahIDR) {

            saldo -= jumlahIDR;

            return true;

        } else {

            System.out.println("Saldo tidak cukup.");

            return false;

        }

    }

}

class TestMultiTabungan {

    public static *void* main(String[] *args*) {

        MultiTabungan rekening = **new** MultiTabungan(1000000);

        rekening.simpanUang("AUD", 10);

        System.out.println("Saldo: " + rekening.getSaldo() + " IDR");

        if (rekening.ambilUang("IDR", 1000000)) {

            System.out.println("Penarikan berhasil.");

        }

        rekening.simpanUang("USD", 10);

        System.out.println("Saldo: " + rekening.getSaldo() + " IDR");

        if (rekening.ambilUang("USD", 10)) {

            System.out.println("Penarikan berhasil.");

        }

    }

}

**Kelas MultiTabungan:**

1. **Variabel Instance**:
   * **saldo**: Variabel yang menyimpan saldo dalam mata uang IDR.
2. **Konstruktor**:
   * Menerima parameter **initsaldo** yang merupakan saldo awal dalam mata uang IDR.
   * Inisialisasi saldo dengan nilai dari parameter **initsaldo**.
3. **Metode getSaldo**:
   * Mengembalikan saldo dalam mata uang IDR.
4. **Metode simpanUang**:
   * Menerima dua parameter, yaitu **mataUang** yang merupakan jenis mata uang (IDR, USD, atau AUD) dan **jumlah** yang merupakan jumlah uang yang akan disimpan.
   * Menambahkan jumlah uang ke saldo berdasarkan mata uang yang dipilih. Konversi mata uang USD dan AUD ke IDR dilakukan sesuai dengan nilai tukar yang telah ditentukan.
5. **Metode ambilUang**:
   * Menerima dua parameter, yaitu **mataUang** yang merupakan jenis mata uang (IDR, USD, atau AUD) dan **jumlah** yang merupakan jumlah uang yang akan ditarik.
   * Mengurangi jumlah uang dari saldo berdasarkan mata uang yang dipilih. Konversi mata uang USD dan AUD ke IDR dilakukan sesuai dengan nilai tukar yang telah ditentukan. Jika saldo mencukupi, penarikan berhasil dan metode mengembalikan **true**, jika tidak, penarikan gagal dan metode mengembalikan **false**.

**Kelas TestMultiTabungan:**

1. **Metode main**:
   * Membuat objek **rekening** dari kelas **MultiTabungan** dengan saldo awal 1.000.000 IDR.
   * Menyimpan 10 AUD ke dalam rekening menggunakan metode **simpanUang**.
   * Menampilkan saldo rekening menggunakan metode **getSaldo**.
   * Mencoba menarik 1.000.000 IDR dari rekening menggunakan metode **ambilUang**. Jika penarikan berhasil, mencetak pesan "Penarikan berhasil.".
   * Menyimpan 10 USD ke dalam rekening menggunakan metode **simpanUang**.
   * Menampilkan saldo rekening menggunakan metode **getSaldo**.
   * Mencoba menarik 10 USD dari rekening menggunakan metode **ambilUang**. Jika penarikan berhasil, mencetak pesan "Penarikan berhasil.".

Program ini mengilustrasikan penggunaan kelas untuk mengelola saldo tabungan dalam beberapa mata uang dan melakukan operasi simpan dan tarik uang.