|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Larastika Putri Hendarti**  **G1F021030** | **Tipe Data** | **07 Oktober 2024** |

**[No. 1] Identifikasi Masalah:**

1.1. Evaluasi penyebab kesalahan terjadi dan perbaiki agar program dapat berjalan!  
1.2. Susun kode program Java yang menampilkan data pribadi Anda! (min. 5 informasi data diri)

**[No.1] Analisis dan Argumentasi**

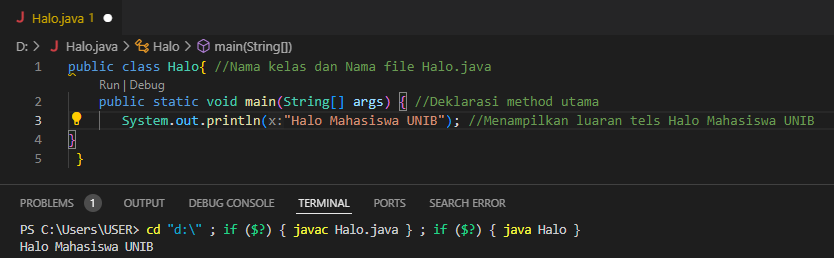
1. Saya mengusulkan permasalahan ini dapat diatasi dengan cara string harus selalu dibuka dan ditutup dengan tanda kutip ganda (")
2. Alasan solusi ini karena Kesalahan penutupan string menyebabkan kompiler tidak dapat memahami di mana string dimulai dan berakhir.
3. Perbaikan kode program dengan cara Menambahkan tanda kutip ganda yang hilang setelah kalimat "Halo Mahasiswa UNIB"

**[No.1 ] Penyusunan Algoritma dan Kode Program**

1. Algoritma

Algoritma adalah langkah-langkah penyelesaian masalah.

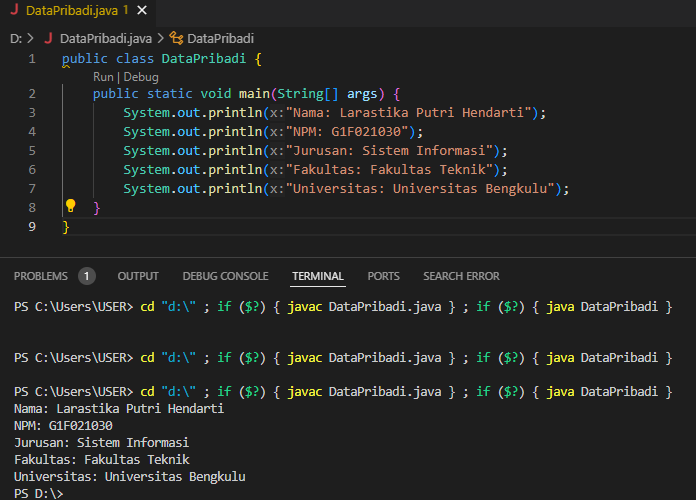
1. Mulai
2. Deklarasikan kelas KelasKu
3. Deklarasikan metode main dengan tanda public static void main(String[] args)
4. Di dalam metode main, panggil perintah System.out.println untuk menampilkan pesan "Halo Mahasiswa UNIB"
5. Tutup blok metode dengan }
6. Tutup blok kelas dengan }
7. Selesai
8. Kode program dan luaran



Diatas merupakan Luaran sesuai dengan program yang disusun.   
Tipe data yang ditampilkan telah sesuai dengan kebutuhan dan permintaan data.

Adapun perbaikan dari kode sebelumnya antara lain :

* Menutup string literal dengan tanda kutip ganda di akhir kalimat.
* Mengubah metode main menjadi public static void main.
* Menambahkan kurung kurawal penutup untuk mengakhiri blok program.

  
Pada gambar diatas merupakan program Java dengan outut yang menampilkan data pribadi saya.

Dengan program ini, kelima data pribadi ditampilkan secara berurutan menggunakan metode System.out.println untuk setiap baris informasi.

**[No.1] Kesimpulan**

Solusi dari permasalahan ini terletak pada memperbaiki sintaks yang salah. Dengan menutup string literal, menggunakan metode main yang tepat, dan melengkapi kurung kurawal, kode dapat dijalankan tanpa error.

**[No. 2] Identifikasi Masalah:**

2.1. Susun kembali sintaks kode untuk memperbaiki pesan kesalahan tersebut! Berikan kesimpulan!

**[No.2] Analisis dan Argumentasi**

1. Saya mengusulkan permasalahan ini dapat diatasi dengan cara menggunakan tipe data yang tepat untuk angka besar (long untuk a dan int untuk b) serta memastikan bahwa karakter hanya berisi satu huruf untuk char.
2. Alasan solusi ini karena Java memiliki batasan ukuran data untuk setiap tipe data primitif. int hanya bisa menyimpan angka hingga 2,147,483,647, sedangkan byte lebih kecil lagi. char hanya bisa menyimpan satu karakter.
3. Perbaikan kode program dengan cara mengganti:

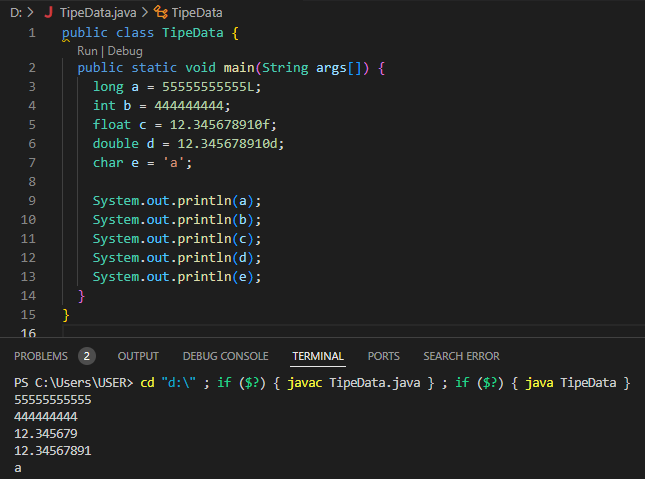
* Tipe data int menjadi long untuk nilai besar.
* Tipe data byte menjadi int untuk nilai lebih kecil.
* Memastikan tipe char hanya berisi satu karakter.

**[No.1 ] Penyusunan Algoritma dan Kode Program**

1. Algoritma

Algoritma adalah langkah-langkah penyelesaian masalah.

1. Mulai
2. Deklarasikan kelas TipeData
3. Deklarasikan metode main dengan tanda public static void main(String[] args)
4. Deklarasikan variabel: long a, int b, float c, double d, dan char e
5. Masukkan nilai yang sesuai untuk masing-masing variabel
6. Gunakan System.out.println untuk menampilkan nilai-nilai variabel
7. Tutup blok metode dan kelas dengan kurung kurawal penutup
8. Selesai
9. Kode program dan luaran



Di atas merupakan luaran sesuai dengan program yang disusun. Tipe data yang ditampilkan telah sesuai dengan kebutuhan dan permintaan data. Adapun perbaikan dari kode sebelumnya antara lain:

* Mengganti int dengan long untuk menangani angka besar.
* Mengubah byte menjadi int.
* Menyesuaikan char agar hanya menyimpan satu karakter.

**[No.2] Kesimpulan**

Solusi dari permasalahan ini terletak pada pemilihan tipe data yang sesuai untuk nilai yang besar dan memperbaiki kesalahan penggunaan char. Dengan menggunakan tipe data yang tepat, kode dapat dijalankan tanpa error.

**[No. 3] Identifikasi Masalah:**

* 1. Salin dan tempel potongan kode ini ke Eclipse atau Jdoodle.

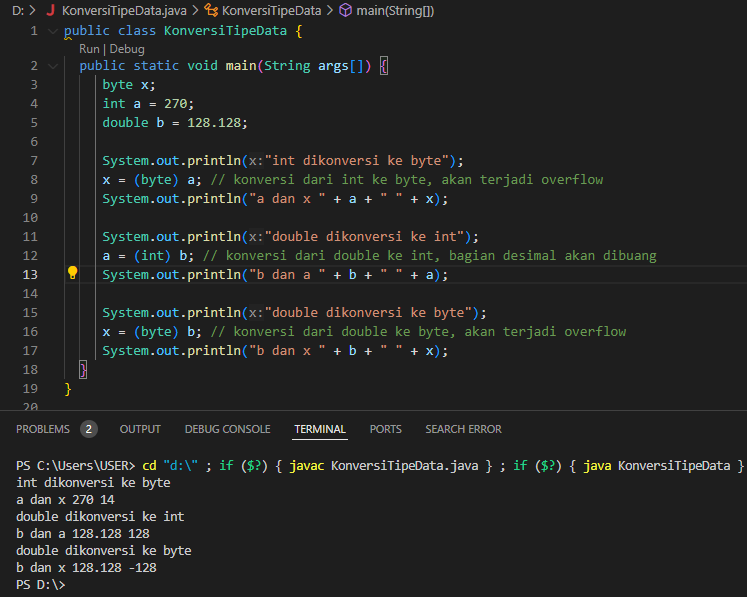
**[No.3] Analisis dan Argumentasi**

1. Saya mengusulkan permasalahan ini dapat diatasi dengan cara menggunakan casting yang sesuai untuk tipe data yang lebih kecil, dan memperhatikan potensi overflow saat konversi nilai yang lebih besar.
2. Alasan solusi ini karena Java mengikuti aturan konversi tipe data secara ketat, dan saat konversi dari tipe data yang lebih besar (misalnya int atau double) ke tipe data yang lebih kecil (seperti byte), nilai dapat mengalami perubahan akibat batas memori yang tersedia.
3. Perbaikan kode program dengan cara memastikan tipe data yang dikonversi tetap berada dalam batas yang sesuai, atau menambahkan logika untuk menangani overflow jika diperlukan.

**[No.3 ] Penyusunan Algoritma dan Kode Program**

Algoritma adalah langkah-langkah penyelesaian masalah.

1. Mulai
2. Deklarasikan kelas KonversiTipeData
3. Deklarasikan metode main dengan tanda public static void main(String[] args)
4. Deklarasikan variabel int a, double b, dan byte x
5. Lakukan konversi tipe data dengan casting secara eksplisit
6. Gunakan System.out.println untuk menampilkan hasil konversi setiap tipe data
7. Tutup blok metode dan kelas dengan kurung kurawal penutup
8. Selesai
9. Kode program dan luaran



Pada kode KonversiTipeData, ada beberapa hal yang perlu diperhatikan terkait konversi tipe data. Salah satu masalah yang muncul adalah:

* Overflow saat konversi dari int atau double ke byte karena nilai melebihi batas yang dapat disimpan oleh tipe data byte.

Perbaikan Kode:

* Lakukan casting secara eksplisit pada saat mengubah tipe data yang lebih besar ke yang lebih kecil.
* Berikan perhatian khusus terhadap overflow saat menggunakan tipe data yang kecil seperti byte.

**[No.3] Kesimpulan**

Solusi dari permasalahan ini terletak pada penggunaan konversi tipe data yang sesuai serta memperhatikan batasan dari setiap tipe data, khususnya saat bekerja dengan tipe data yang kecil seperti byte. Dengan menggunakan casting dan menyadari adanya overflow, kode dapat dijalankan tanpa error dan menghasilkan luaran yang diharapkan.

**[No. 4] Identifikasi Masalah:**

4.1. Konstruksikan kode program konversi tipe data pada Latihan 3 ke bentuk tipe data lain yang kompatibel.

4.2. Simpulkan alasan kode yang disusun, jenis konversi tipe data implisit dan eksplisit!

**[No.3] Analisis dan Argumentasi**

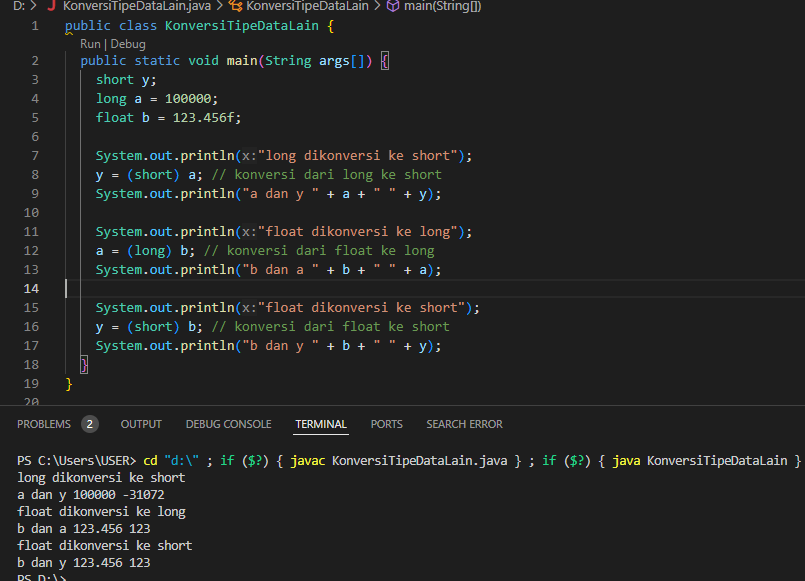
1. Saya mengusulkan permasalahan ini dapat diatasi dengan cara melakukan casting eksplisit pada saat konversi dari tipe data yang lebih besar (seperti long dan float) ke tipe data yang lebih kecil (seperti short). Selain itu, perlu memperhatikan kemungkinan overflow.
2. Alasan solusi ini karena konversi dari tipe data yang lebih besar ke tipe data yang lebih kecil (seperti dari float ke short) dapat menyebabkan perubahan nilai yang tidak diinginkan akibat keterbatasan tipe data yang lebih kecil. Casting eksplisit diperlukan agar program dapat tetap berjalan dengan baik dan hasilnya dapat diprediksi.
3. Perbaikan kode program dilakukan dengan cara menggunakan casting eksplisit untuk semua konversi dari tipe data besar ke tipe data kecil, seperti:

* y = (short) a untuk konversi dari long ke short.
* a = (long) b untuk konversi dari float ke long.

**[No.3 ] Penyusunan Algoritma dan Kode Program**

Algoritma adalah langkah-langkah penyelesaian masalah.

1. Mulai
2. Deklarasikan kelas KonversiTipeDataLain
3. Deklarasikan metode main dengan tanda public static void main(String[] args)
4. Deklarasikan variabel long a, float b, dan short y
5. Gunakan casting eksplisit untuk mengonversi nilai long ke short, dan float ke long dan short
6. Gunakan System.out.println untuk menampilkan hasil konversi
7. Tutup blok metode dan kelas dengan kurung kurawal penutup
8. Selesai
9. Kode program dan luaran



**[No.3] Kesimpulan**

Solusi dari permasalahan ini terletak pada melakukan casting secara eksplisit ketika tipe data yang lebih besar harus dikonversi ke tipe data yang lebih kecil. Konversi semacam ini harus dilakukan dengan hati-hati, karena bisa mengakibatkan overflow yang mengubah nilai asli. Dengan casting yang benar, kode dapat dijalankan tanpa error, dan hasil konversi dapat diatur sesuai dengan ekspektasi.

**Refleksi**

Pada minggu ini, saya mempelajari lebih lanjut tentang konversi tipe data, terutama tentang overflow yang terjadi ketika melakukan konversi dari tipe data besar ke tipe data kecil. Tantangan terbesar adalah memahami bagaimana Java menangani overflow secara otomatis dan kapan casting eksplisit diperlukan.