JobSheet - Week 1 - Fundamental Programming Structures in Java

Nama: Faqih Mujaddid

NIM: 251524096

Kelas: 1D

Repo GitHub: https://github.com/faqihmujaddid/Teknin-Pemrograman/tree/main

Instruksi Pengerjaan:

1. Kerjakan 5 soal di bawah ini dengan melengkapi setiap kolom jawaban yang disediakan pada jobsheet ini.
2. Jawaban setiap soal mencakup source code, screenshot hasil dari program yang ditampilkan full screen termasuk taskbar (tambahkan beberapa screenshot jika diperlukan), penjelasan permasalahan dan solusi yang dihadapi, nama teman yang membantu memecahkan masalah (opsional).
3. Dikumpulkan pada Assignment Classroom sesuai dengan deadline yang tertera pada assignment tersebut.
4. Format penamaan file jobsheet: W1_P_<Kelas 1X>_<3 Digit_NIM_Terakhir>.docx/pdf. Contoh: W1_P_1B_001.docx/pdf.
5. Buatlah satu file java yang mengandung jawaban dalam bentuk source untuk satu jawaban yang dapat langsung dieksekusi. Contoh penamaan: 1-DataTypes.java, 2-Variables.java, 3-Arithmetic.java, 4-TypeCasting.java, dan 5-Operator.java.
6. Submit semua jawaban dalam bentuk file java pada repository GitHub masing-masing.

No. 1 Data Types

Soal Praktikum
Java memiliki 8 tipe data primitif; char, boolean, byte, short, int, long, float, dan double. Untuk praktikum ini, kita akan Latihan dengan tipe data primitif yang digunakan untuk menyimpan nilai bilangan bulat, yaitu byte, short, int, dan long. • A byte is an 8-bit signed integer. • A short is a 16-bit signed integer. • An int is a 32-bit signed integer. • A long is a 64-bit signed integer. Dengan diberikan sebuah bilangan bulat masukan, Anda harus menentukan tipe data primitif mana yang mampu menyimpan masukan tersebut dengan benar.

Input Format
Baris pertama berisi bilangan bulat, T, yang menunjukkan jumlah kasus uji. Setiap kasus uji, T, terdiri dari satu baris dengan bilangan bulat, n, yang nilainya bisa sangat besar atau sangat kecil.
Output Format
Untuk setiap variabel masukan n dan tipe data primitif yang sesuai, Anda harus menentukan apakah tipe data primitif yang diberikan mampu menyimpannya. Jika ya, maka cetak:

```
N can be fitted in:
* datatype
```

Jika terdapat lebih dari satu tipe data yang sesuai, cetak masing-masing pada barisnya sendiri dan urutkan berdasarkan ukurannya (misalnya: **byte < short < int < long**).
Jika angka tersebut tidak dapat disimpan dalam salah satu dari empat tipe data primitif yang disebutkan di atas, cetak baris berikut:

```
N can’t be fitted anywhere
```

Sample Input:

```
5
-150
150000
1500000000
213333333333333333333333333333333333
-1000
```

Sample Output:

```
-150 can be fitted in:
  *  short
  *  int
  *  long
150000 can be fitted in:
  *  int
  *  long
1500000000 can be fitted in:
  *  int
  *  long

213333333333333333333333333333333333 can't be fitted anywhere.
```

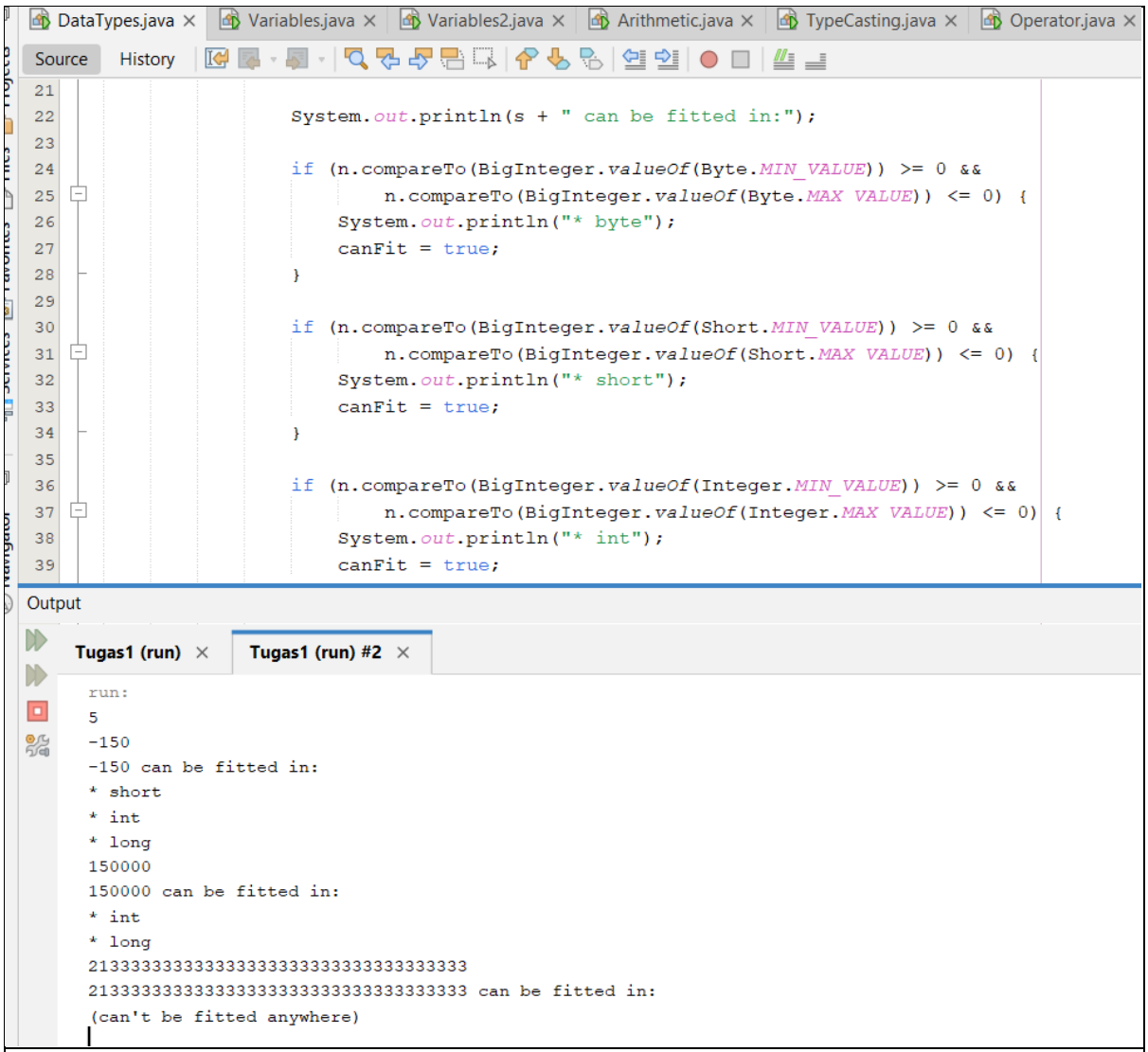
Explanation:
Angka 150 dapat disimpan dalam tipe data short, int, atau long. Angka 213333333333333333333333333333333333 sangat besar dan berada di luar rentang nilai yang diizinkan untuk tipe data primitif yang dibahas dalam masalah ini.

Source Code

```
package pkg1.datatypes;
import java.util.Scanner;
import java.math.BigInteger;

public class DataTypes {
    public static void main(String[] args) {
```

```
        try (Scanner sc = new Scanner(System.in)) {
            int T = sc.nextInt();

            for (int i = 0; i < T; i++) {
                String s = sc.next();

                if (!s.matches("[+-]?\\d+")) {
                    System.out.println(s + " can't be fitted anywhere.");
                    continue;
                }

                BigInteger n = new BigInteger(s);

                boolean canFit = false;

                System.out.println(s + " can be fitted in:");

                if (n.compareTo(BigInteger.valueOf(Byte.MIN_VALUE)) >= 0 &&
                        n.compareTo(BigInteger.valueOf(Byte.MAX_VALUE)) <=
0) {
                    System.out.println("* byte");
                    canFit = true;
                }

                if (n.compareTo(BigInteger.valueOf(Short.MIN_VALUE)) >= 0
&&
                        n.compareTo(BigInteger.valueOf(Short.MAX_VALUE)) <=
0) {
                    System.out.println("* short");
                    canFit = true;
                }

                if (n.compareTo(BigInteger.valueOf(Integer.MIN_VALUE)) >= 0
&&
                        n.compareTo(BigInteger.valueOf(Integer.MAX_VALUE))
<= 0) {
                    System.out.println("* int");
                    canFit = true;
                }

                if (n.compareTo(BigInteger.valueOf(Long.MIN_VALUE)) >= 0 &&
                        n.compareTo(BigInteger.valueOf(Long.MAX_VALUE)) <=
0) {
                    System.out.println("* long");
                    canFit = true;
                }


                if (!canFit) {

                    System.out.println("(can't be fitted anywhere)");
                }
            }
        }
    }
}
```

Screenshot Hasil



Penjelasan Permasalahan dan Solusi

Setelah mencoba sendiri, saya akhirnya mulai memahami logikanya. Namun, saya masih perlu mencari referensi perintah (command) di W3Schools dan di bantu oleh teman saya agar bisa mengimplementasikannya dengan benar. Setelah mengikuti contoh yang ada, program akhirnya berhasil dijalankan.

Akan tetapi, muncul masalah ketika mencoba beberapa angka pada sample output. Tidak semua angka bisa diproses karena terjadi overflow. Setelah mencari tahu lebih lanjut, saya memahami bahwa nilai yang digunakan sudah melebihi batas kapasitas bit pada tipe data primitif.

Untuk mengatasi masalah tersebut, saya kembali mencari solusi di website yang sama dan menemukan penggunaan BigInteger. Setelah menerapkan BigInteger pada program, permasalahan overflow berhasil diatasi dan program dapat berjalan dengan baik.

Nama Teman Hal yang Dibantu (Opsional)

Mohamad Haafiz Mawla Nayyara

No. 2 Variables

Soal Praktikum

Perhatikan dua bagian program di bawah ini.

Bagian 1:

```
public class Constants {
	public static void main(String[] args) {
		final double CM_PER_INCH = 2.54; double paperWidth = 8.5;
		double paperHeight = 11;
		System.out.println("Paper size in centimeters: " + paperWidth
* CM_PER_INCH + " by " + paperHeight * CM_PER_INCH);
	}
}
```

Bagian 2:

```
public class Constants2 {
	public static final double CM_PER_INCH = 2.54; public static void
main(String[] args) {
		double paperWidth = 8.5; double paperHeight = 11;
		System.out.println("Paper size in centimeters: " + paperWidth
* CM_PER_INCH + " by " + paperHeight * CM_PER_INCH);
	}
}
```

Dari 2 contoh baris program diatas, jawablah pertanyaan dibawah ini:

1. Bagaimana output dari masing masing class Constants dan Constants2?
2. Apa perbedaan penggunaan final double dengan public static final double?

Source Code

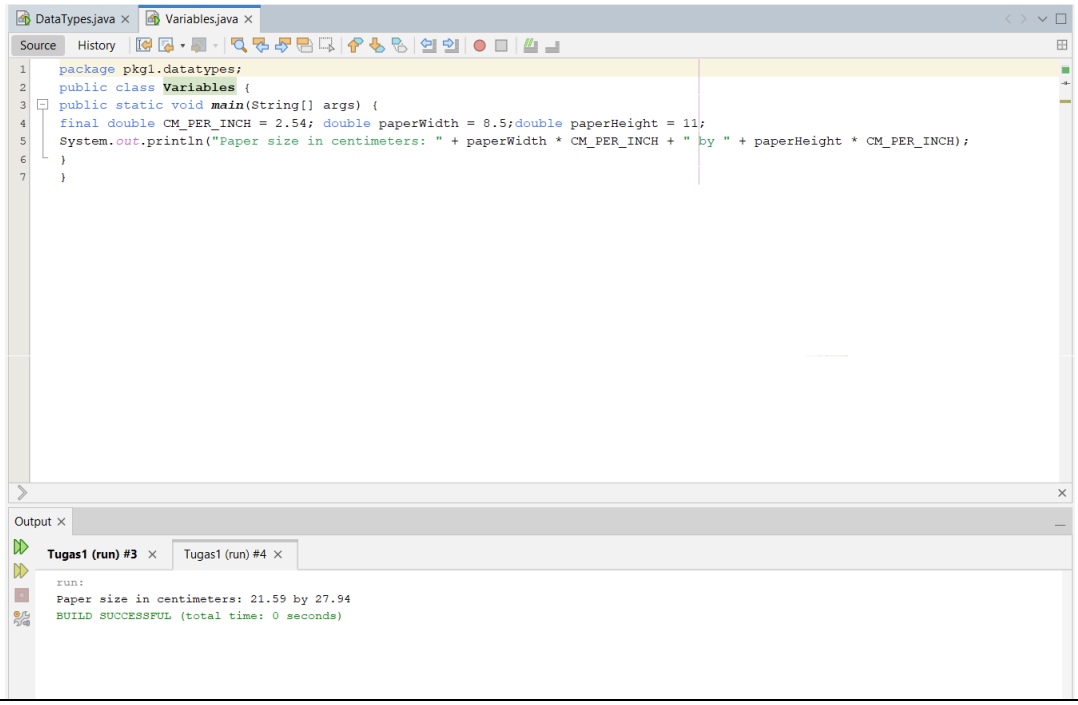
Bagian 1:

```
public class Variables {
	public static void main(String[] args) {
		final double CM_PER_INCH = 2.54; double paperWidth = 8.5;
		double paperHeight = 11;
		System.out.println("Paper size in centimeters: " + paperWidth
* CM_PER_INCH + " by " + paperHeight * CM_PER_INCH);
	}
}
```

Bagian 2:

```
public class Variables2 {
	public static final double CM_PER_INCH = 2.54; public static void
main(String[] args) {
		double paperWidth = 8.5; double paperHeight = 11;
		System.out.println("Paper size in centimeters: " + paperWidth
* CM_PER_INCH + " by " + paperHeight * CM_PER_INCH);
	}
}
```

Screenshot Hasil



Penjelasan Permasalahan dan Solusi

Karena saya belum pernah menggunakan java saya sedikit kesulitan saat pertama kali menggunakanya seperti apa itu pacage dan apa itu public class, namun teman saya yang sudah cukup mahir menggunakan java memberi tahu saya caranya dan menjadi faham

Nama Teman dan Hal yang Dibantu (Opsional)

Mohamad Haafiz Mawla Nayyara

No. 3 Arithmetic - Math Class

Soal Praktikum

Perhatikan bagian program di bawah ini.

```
Class FloatingPoint {
	public static void main(String[] args) {
		double x = 92.98;
		int nx = (int) Math.round(x);
	}
```

```
}
```

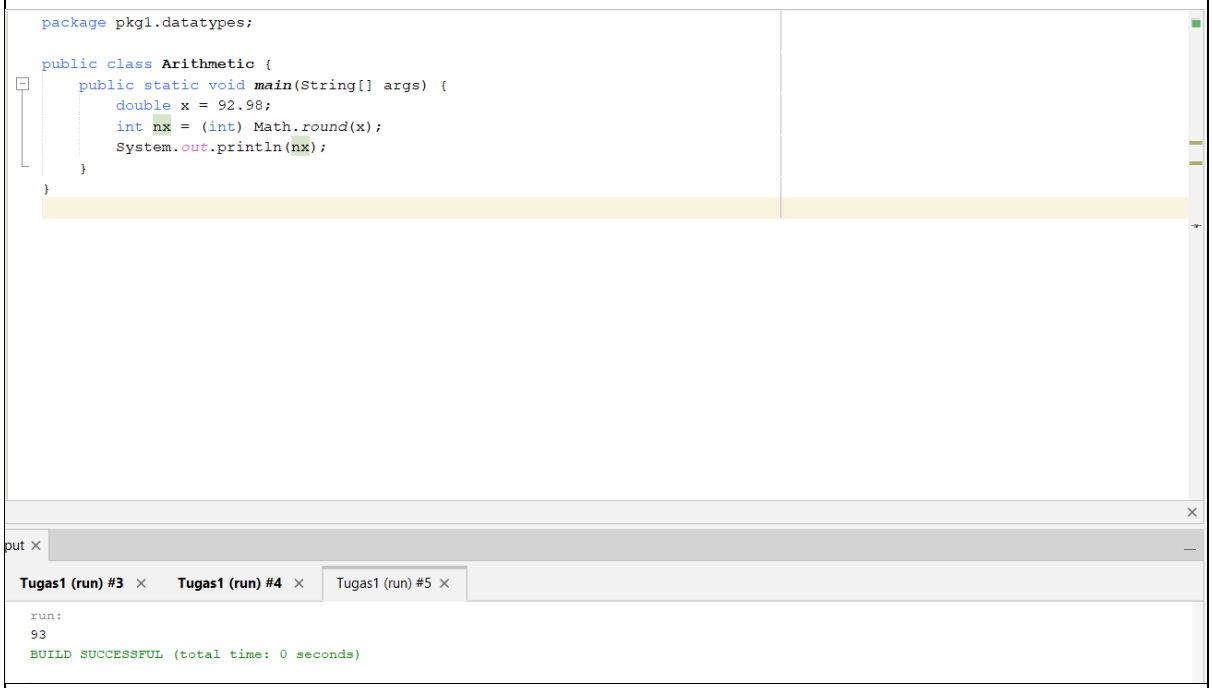
Math Class berisi bermacam-macam fungsi matematika seperti pada contoh diatas pada penggunaan round(x), terdapat beberapa pertanyaan yang perlu untuk dijelaskan:

1. Pada kasus berikut jelaskan nilai nx setelah digunakan **Math.round(x)**!
2. Kenapa dibutuhkan cast (int) dalam penggunaan **Math.round(x)**?

Source Code

```
Class FloatingPoint {
	public static void main(String[] args) {
		double x = 92.98;
		int nx = (int) Math.round(x);
		System.out.println(nx);
 }
}
```

Screenshot Hasil



Penjelasan Permasalahan dan Solusi

Program tidak mengeluarkan hasil apapun di karenakan program belum memiliki komentar output sehingga tidak memunculkan hasil apapun dan saya menamahkan output dengan contoh dari no 1

1.Nilai nx setelah Math.round(x)
Math.round(92.98) membulatkan ke bilangan bulat terdekat → 93
Jadi nx akhirnya 93 (yang dicetak juga 93).

2.Karena Math.round(x) untuk tipe double mengembalikan long, bukan int.
Sedangkan nx bertipe int, jadi harus diubah dulu dari long ke int dengan cast (int).

Nama Teman dan Hal yang Dibantu (Opsional)

Firon Fariz Sahuleka

No. 4 Type Casting/ Data Type Conversion

Soal Praktikum

Perhatikan baris program dibawah ini:

```
class ConvertDataType {
	static short methodOne(long l) {
		int i = (int) l; return (short)i;
	}

	public static void main(String[] args) {
		double d = 10.25; float f = (float) d;
		byte b = (byte) methodOne((long) f); System.out.println(b);
	}
}
```

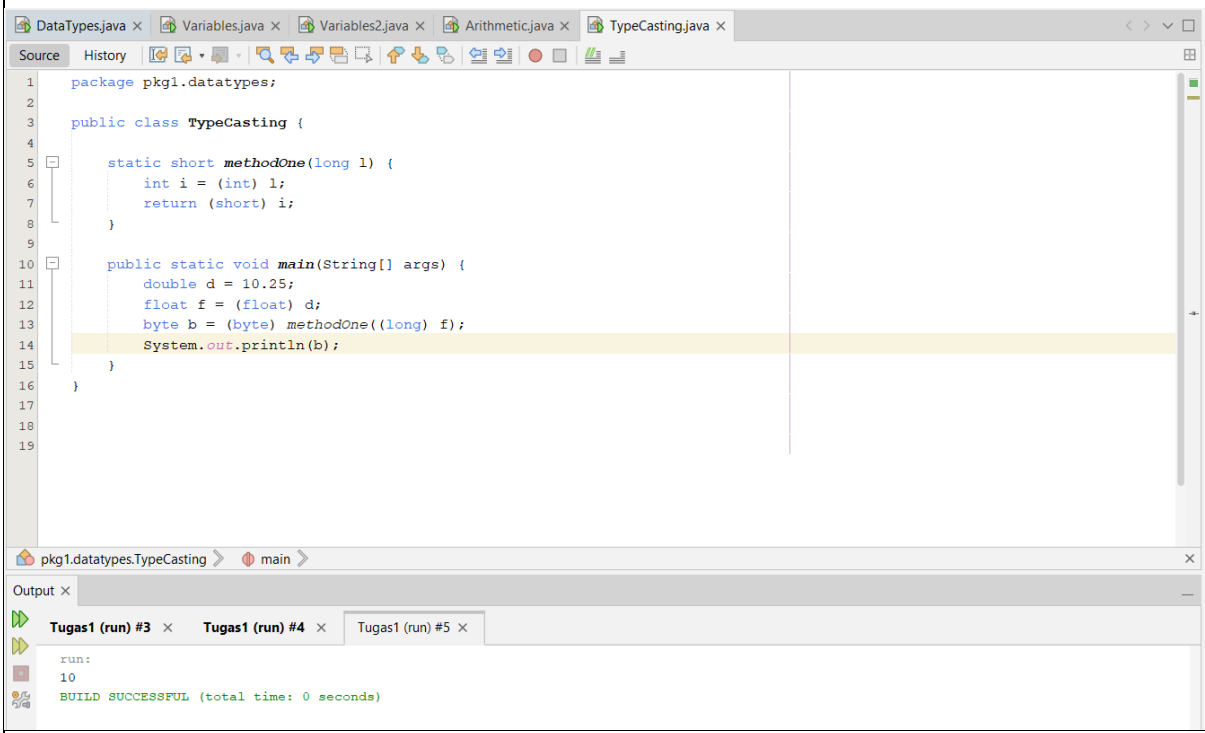
Program berikut melakukan convert tipe data yang berukuran besar ke kecil (long -> int -> short) dan (double -> float -> byte).

1. Jelaskan output nilai dari variable b.
2. Jelaskan apa yang berubah dari variable d menjadi variable b setelah dilakukan cast?

Source Code

```
package pkg1.datatypes;

public class TypeCasting {

    static short methodOne(long l) {
        int i = (int) l;
        return (short) i;
    }

    public static void main(String[] args) {
        double d = 10.25;
        float f = (float) d;
        byte b = (byte) methodOne((long) f);
        System.out.println(b);
    }
}
```

Screenshot Hasil



Penjelasan Permasalahan dan Solusi

hapus class ConvertDataType karena sebenarnya tidak perlu dan

1. Output b = 10
Karena: d = 10.25 → jadi float 10.25 → di-cast ke long jadi **10** (di bulatkan kebawah jika desimal dibawah 5 dan ke atas jika desimal di atas 5) → turun lagi ke int/short/byte tetap 10 (masih dalam batas).
2. Tipe data mengecil: double → float → long → int → short → byte (range makin kecil), tapi nilainya tetap 10 karena tidak overflow.

Nama Teman dan Hal yang Dibantu (Opsional)

No. 5 Operator

Soal Praktikum

Perhatikan bagian program di bawah ini.

```
class OperatorChallenge {
    public static void main(String[] args) {
        int a = 5;
        int b = 10;

        boolean result = (++a * 2 > b) && (b++ % 3 == 1);

        System.out.println("Hasil Boolean: " + result);
        System.out.println("Nilai a: " + a);
        System.out.println("Nilai b: " + b);
    }
}
```

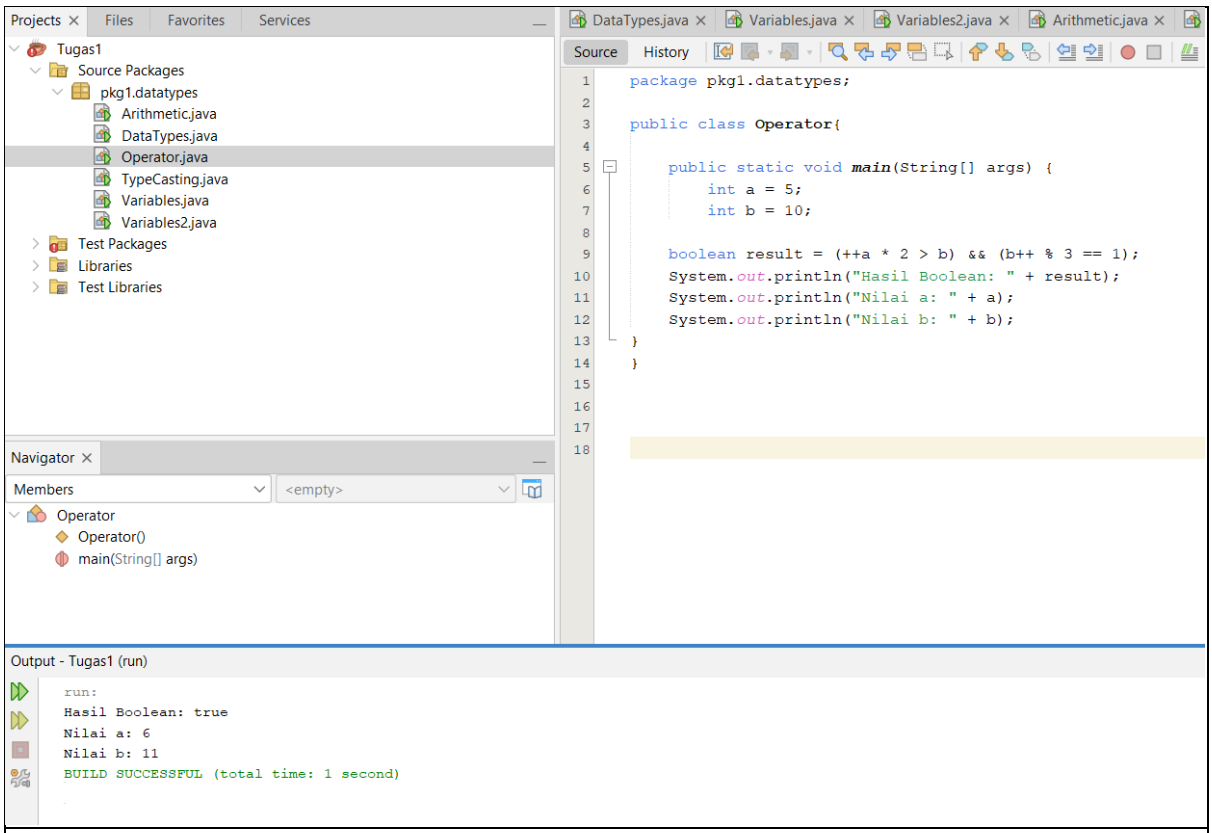
Pertanyaan Analisis:

1. **Analisis Langkah Demi Langkah:** Jelaskan urutan eksekusi pada baris `boolean result`. Mana yang dijalankan lebih dulu antara `++a` dan perkalian `*`?
2. **Short-Circuit Logic:** Jika bagian pertama `(++a * 2 > b)` bernilai `false`, apakah bagian kedua `(b++ % 3 == 1)` akan tetap dieksekusi oleh Java? Jelaskan dampaknya pada nilai akhir variabel `b`.
3. **Output:** Berapakah nilai akhir dari `result`, `a`, dan `b`?

Source Code

```
package pkg1.datatypes;

public class Operator{

    public static void main(String[] args) {
        int a = 5;
        int b = 10;

    boolean result = (++a * 2 > b) && (b++ % 3 == 1);
    System.out.println("Hasil Boolean: " + result);
    System.out.println("Nilai a: " + a);
    System.out.println("Nilai b: " + b);
}
}
```

Screenshot Hasil



Penjelasan Permasalahan dan Solusi

Untuk no ini saya belum menemukan masalah dikarenakan sesuai dengan source code yang diberi

1. Pada baris boolean result = (++a * 2 > b) && (b++ % 3 == 1);, yang dijalankan lebih dulu adalah ++a. Nilai a awalnya 5, lalu karena menggunakan pre-increment (++a), maka a langsung menjadi 6 sebelum dikalikan. Setelah itu dilakukan perkalian 6 * 2 = 12, lalu dibandingkan dengan b yang bernilai 10. Karena 12 > 10, maka bagian pertama bernilai true.

2. Karena menggunakan operator &&, jika bagian pertama bernilai false, maka bagian kedua tidak akan dijalankan (short-circuit). Artinya b++ tidak akan dieksekusi dan nilai b tidak berubah. Namun pada kasus ini bagian pertama bernilai true, sehingga bagian kedua tetap dijalankan.

3. Pada bagian kedua, b++ menggunakan nilai lama terlebih dahulu yaitu 10, sehingga 10 % 3 = 1 dan hasil perbandingan 1 == 1 adalah true. Setelah itu barulah b bertambah menjadi 11. Jadi nilai akhirnya adalah result = true, a = 6, dan b = 11.

Nama Teman dan Hal yang Dibantu (Opsional)