JobSheet - Week 1 - Fundamental Programming Structures in Java

Nama : **Raza Mulki Firdaus**

NIM : **251524120**

Kelas : **1D**

Repo GitHub : **https://github.com/zaaraza/Teknik-Pemrograman-**

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
21333333333333333333333333333333333333
-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

21333333333333333333333333333333333333 can't be fitted anywhere.
```

Explanation:
Angka 150 dapat disimpan dalam tipe data short, int, atau long. Angka 21333333333333333333333333333333333333 sangat besar dan berada di luar rentang nilai yang diizinkan untuk tipe data primitif yang dibahas dalam masalah ini.

Source Code

```java
package Pertemuan1;

import java.util.Scanner;
import java.math.BigInteger;

public class Cobainlagi {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int T = sc.nextInt();
        for (int i = 0; i < T; i++) {
            String input = sc.next();
            BigInteger n = new BigInteger(input);

            BigInteger minLong = BigInteger.valueOf(Long.MIN_VALUE);
            BigInteger maxLong = BigInteger.valueOf(Long.MAX_VALUE);

            if (n.compareTo(minLong) < 0 || n.compareTo(maxLong) > 0) {
                System.out.println(input + " can't be fitted anywhere.");
                continue;
            }

            System.out.println(input + " 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");
            }

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

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

            System.out.println("* long");
        }

        sc.close();
    }
}
```

Screenshot Hasil

```
5
-150
-150 can be fitted in:
* short
* int
* long
150000
150000 can be fitted in:
* int
* long
1500000000
1500000000 can be fitted in:
* int
* long
213333333333333333333333333333333333
213333333333333333333333333333333333 can't be fitted anywhere.
-1000
-1000 can be fitted in:
* short
* int
* long
```

Penjelasan Permasalahan dan Solusi

Soal ini mirip seperti soal CP dimana terdapat narasi mengenai tipe data pada Java, harus menentukan apakah sebuah bilangan bulat dapat disimpan dalam tipe data primitif Java yaitu byte, short, int, atau long. Program menerima jumlah input sebanyak T, dan user menginput angka sebanyak T kali yang kemudian program akan menghasilkan output bilangan yang diinputkan bisa termasuk tipe data apa saja (byte, short, int, atau long).

Solusinya pada source code adalah kita buat dulu code untuk user menginput banyaknya jumlah uji kasus menggunakan Scanner, lalu setiap angka dibaca dalam bentuk string. Setelah itu string diubah menjadi BigInteger supaya bisa dibandingkan tanpa batas ukuran seperti pada tipe primitif. Setelah itu gunakan algoritma kondisional yaitu if untuk mengecek angka inputan user termasuk tipe data apa

saja. Jika nilainya berada di luar rentang long, program langsung mencetak bahwa angka tidak bisa disimpan di mana pun. Jika masih dalam rentang long, program mengecek satu per satu mulai dari byte hingga long dan mencetak tipe data yang sesuai secara berurutan dari yang paling kecil ke yang paling besar.

Nama Teman Hal yang Dibantu (Opsional)
-

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

```java
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);
    }
}
```

Screenshot Hasil

Contstans

```
Paper size in centimeters: 21.59 by 27.94
```

Constans 2

```
Paper size in centimeters: 21.59 by 27.94
```

Penjelasan Permasalahan dan Solusi

Pada soal diatas ada dua source code berbeda untuk mencari ukuran sebuah kertas,

1. Hasil kedua source code untuk menghitung ukuran kertas adalah sama, sama-sama outputnya adalah Paper in centimeters: 21.59 by 27.94
2. Perbedaannya adalah konstantanyaa. Final double pada source code Constants merupakan konstanta lokal sementara public final doube pada source code Constants2 merupakan konstanta global, konstanta lokal terbatas hanya pada satu class saja, sedangkan konstanta global bisa digunakan oleh class lain.

Nama Teman dan Hal yang Dibantu (Opsional)

-

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

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

Screenshot Hasil

```
x = 92.98
nx = 93
```

```
x = 92.18
nx = 92
```

Penjelasan Permasalahan dan Solusi

1. Pada kasus ini, nilai nx setelah digunakan Math.round(x) adalah 93. Ini berarti Math.round(x) berfungsi untuk membulatkan ke bilangan integer terdekat. Ini terbukti setelah saya mencoba mengganti bilangan 92.98 menjadi 92.18 dan hasilnya nilai nx bukan 93 melainkan 92
2. Karena nx integer sehingga dibutuhkan cast (int) Math.round(x), selain itu juga ketika cast (int) dihilangkan, maka terjadi error pada program. Jadi cast ini mengubah hasil dari long menjadi int agar sesuai dengan tipe variabel nx.

Nama Teman dan Hal yang Dibantu (Opsional)

-

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

```java
public class ConvertDataType {
    public 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

```
10
```

Penjelasan Permasalahan dan Solusi

Code diatas merupakan program untuk melakukan convert tipe data dari besar ke kecil (dari long ke short dan dari double ke byte), setelah diketahui double d = 10.25, hasil convert ke byte nya adalah 10

1. Output nilai dari variabel b adalah 10,
2. Penjelasan
   d = 10.25; float d nilainya tetap lalu saat memanggil methodOne, terjadi cast (long) sehingga (long)f menjadi 10. Saat masuk methodOne parameternya adalah l, jadi long l = 10, dan direturnkan sebagai short 10. Kembali ke main byte b = (byte) methodOne((long)f0 dan nilainya adalah 10. Setelah itu dicast lagi (byte) dan nilainya tetap 10 dan di outputkan

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`?

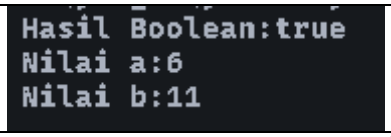
Source Code

```java
public 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);
    }
}
```

Screenshot Hasil



Penjelasan Permasalahan dan Solusi

1. Analisis pada baris boolean result
   Nilai awal : a = 5 ; b =10
   (++a * 2 > b) && (b++ % 3 == 1) yang bernilai true disini berarti operasi dilakukan paling depan dahulu, ++a disana membuat a menjadi 6 dan artinya ++a lah yang dikerjakan dahulu karena terlihat pada operasi boolean tersebut bernilai true saat dikalikan 2 (12 > 10) setelah itu baru mengecek yang belakang/kanan dimana b disini dioperasikan dahulu dengan %3 sehingga menghasilkan true (10%3), setelah itu b ditambah 1.
2. Jika operasi pertama bernilai false, maka Java tidak akan menjalankan operasi setelahnya, karena logika && itu akan bernilai true saat keduanya true, jika salah satu sudah false maka program tidak akan menjalakan operasi setelahnya, dampaknya nilai b akan tetap seperti sebelumnya karena tidak ada increment yang dikerjakan
3. Output nilai akhir adalah a berubah menjadi 6 karena pre increment dan b berubah menjadi 11 karena post increment. Resultnya bernilai true karena kedua kondisi pada operasi boolean sama-sama bernilai true

Nama Teman dan Hal yang Dibantu (Opsional)

-