JobSheet - Week 1 - Fundamental Programming Structures in Java

Nama : Ivanka Bunga Aulia

NIM : 251524105

Kelas : 1D

Repo GitHub: https://github.com/IvankaBunga/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
213333333333333333333333333333333333
-100000000000000
```

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.

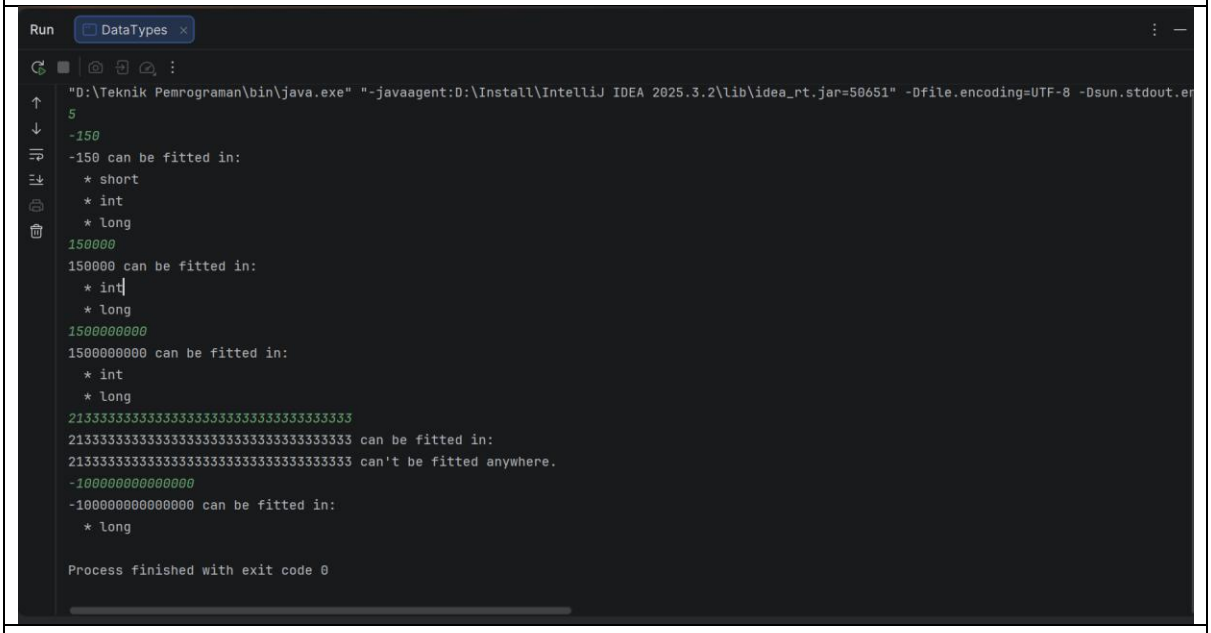
Source Code

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

```
public class DataTypes {

   public static void main(String[] args) {

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

      for (int i = 0; i < T; i++) {

         BigInteger n = sc.nextBigInteger();

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

         boolean canFit = false;

         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(n + " can't be fitted anywhere.");
         }
      }

      sc.close();
```

```
  }
}
```

Screenshot Hasil



Penjelasan Permasalahan dan Solusi

Permasalannya terjadi jika input angka yang sangat besar melebihi rentang tipe data long akan terjadi error, maka solusi yang diberikan adalah dengan menggunakan tipe data input BigInteger agar mampu menyimpan bilangan dalam ukuran yang besar tanpa batas. Setelah mengganti tipe data menjadi BigInteger, seluruh input angka dapat dibaca dengan aman lalu dapat dibandingkan dengan batas maksimum dan minimum masing-masing tipe data menggunakan metode compareTo().

Nama Teman Hal yang Dibantu (Opsional)

M. Haafiz 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);
      }
}
```

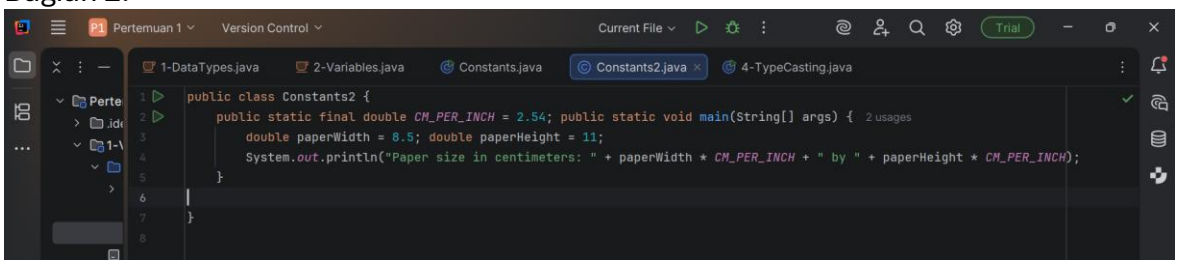
Dari 2 contoh baris program diatas, jawablah pertanyaan dibawah ini:

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

Source Code

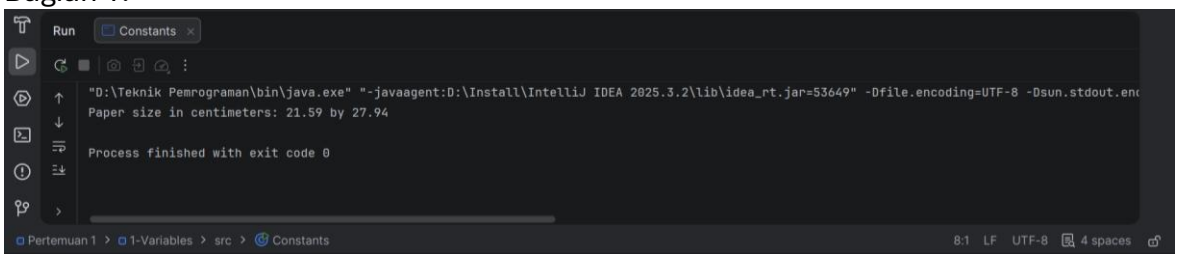
Bagian 1:

Bagian 2:

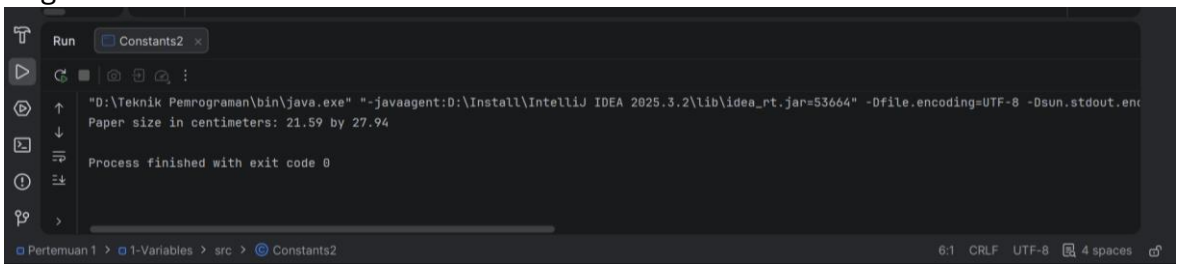


Screenshot Hasil

Bagian 1:



Bagian 2:



Penjelasan Permasalahan dan Solusi

1. Output dari bagian 1 dan bagian 2 sama, yaitu Paper size in centimeters: 21.59 by 27.94

2. Perbedaan antara final double dengan public static final double:
Final double untuk mendeklarasikan konstanta lokal karena berada di dalam main dan hanya bisa digunakan di dalam method tersebut. Lalu konstanta ini juga tidak bisa diakses oleh method atau class baru dan tidak bisa diubah karena posisinya di lokal.

Public static final double karena ada public, maka bisa diakses oleh class lain dan berlaku sebagai konstanta global.

Nama Teman dan Hal yang Dibantu (Opsional)

Syadida Tsaqifa Nanda – Diskusi

No. 3 Arithmetic - Math Class

Soal Praktikum

Perhatikan bagian program di bawah ini.

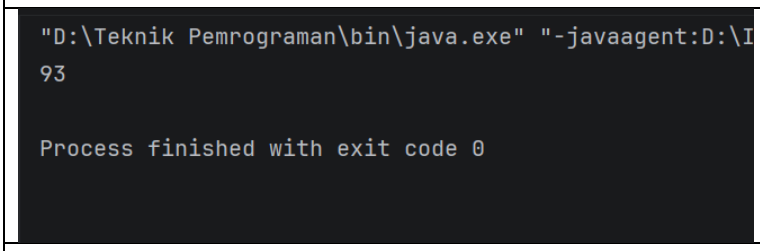
```
Class FloatingPoint {
      public static void main(String[] args) {
            double x = 92.98;
            int nx = (int) Math.round(x);
      }
}
```

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

Menambahkan System.*out*.println(nx); agar terlihat hasilnya

Screenshot Hasil

Penjelasan Permasalahan dan Solusi
1. Nilai x yang tadinya 92.98 setelah menggunakan Math.round(x) menjadi nilai nx = 93. Ini dikarenakan Math.round(x) membulatkan ke bilangan bulat terdekat. 2. Karena math.round(x) mengembalikan tipe long, sedangkan variable nx bertipe integer. Ini dilakukan untuk memperkecil kapasitas penyimpanan. Konversi dari long ke int disebut juga narrowing conversion (dari besar ke kecil).
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

Screenshot Hasil

Penjelasan Permasalahan dan Solusi

jadi di code program ini mengubah tipe data primitif dari double (menyimpan pecahan) menjadi long (menyimpan bilangan bulat).

1. output dari variable nilai b adalah 10

2. variable d berisi tipe data double yang terisi nilai 10.25, lalu setelah dilakukan cast berubah menjadi bilangan bulat tipe data long yang mana menghasilkan variable b dengan nilai 10

Nama Teman dan Hal yang Dibantu (Opsional)

Syadida Tsaqifa Nada - Diskusi

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);
    }
```

}

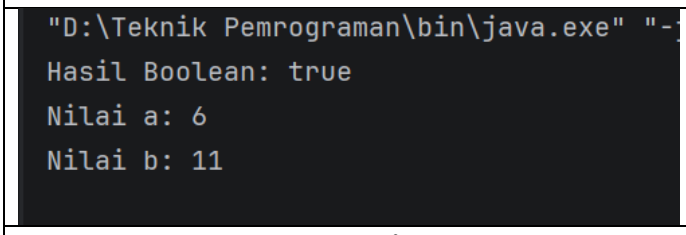
Pertanyaan Analisis:

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

```
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. Langkah awal pada baris Boolean result bagian kiri adalah prefix increment, yaitu menambahkan +1 kedalam variable a menjadi 6. Yang kedua adalah perkalian, ++a * 2 menghasilkan 12 tetapi ini tidak menyimpan pada variable a, hanya hasil sementara. Untuk hasil bagian kiri adalah True.
Sedangkan Langkah awal pada baris Boolean result bagian kanan adalah post-increment, lalu hitung sisa bagi (%) apakah sama dengan 1, jika iya maka True.
Jadi, yang dijalankan terlebih dahulu adalah ++a.

2. Jika bagian kiri false, maka bagian kanan tidak akan dieksekusi. Akibatnya b++ tidak akan dijalankan dan nilai variable b tidak akan berubah, hasil dari boolean juga false. Tetapi jika bagian kiri True, maka bagian kanan akan dieksekusi. Result akan True karena True dan True menghasilkan True.

3. Hasil Boolean: true
Nilai a: 6
Nilai b: 11

Nama Teman dan Hal yang Dibantu (Opsional)