**Laporan Pengerjaan Praktikum Pertemuan 2 Fundamental Programming Structures in Java**

****

**Disusun oleh :**

**Astria Rizka Latifahsary (231524037)**

**Kelas :**

**D4 – 1B Teknik Informatika**

**Tahun Ajaran 2023 – 2024**

Soal 1

Data Types

Java has 8 primitive data types; char, boolean, byte, short, int, long, float, and double. For this  
exercise, we'll work with the primitives used to hold integer values (byte, short, int, and 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.

Given an input integer, you must determine which primitive data types are capable of properly  
storing that input.

Input Format

The first line contains an integer, T, denoting the number of test cases. Each test case, T, is  
comprised of a single line with an integer, n, which can be arbitrarily large or small.

Output Format

For each input variable n and appropriate primitive datatype, you must determine if the given  
primitives are capable of storing it. If yes, then print:

|  |
| --- |
| N can be fitted in:  \* datatype |

If there is more than one appropriate data type, print each one on its own line and order them  
by size (i.e.: ***byte*** < ***short*** < ***int*** < ***long***).

If the number cannot be stored in one of the four aforementioned primitives, print the line:

|  |
| --- |
| 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.  -100000000000000 can be fitted in:  \* long |

**Explanation:**

150 can be stored in a short, an int, or a long.

213333333333333333333333333333333333 is very large and is outside of the allowable range of values for the primitive data types discussed in this problem.

|  |
| --- |
| import java.util.ArrayList;  import java.util.Arrays;  import java.util.List;  public class Main {  public static void main(String[] args) {  int T = Integer.parseInt(System.console().readLine());  for (int i = 0; i < T; i++) {  long n = Long.parseLong(System.console().readLine());  System.out.println(findFittingDatatypes(n));  }  }  public static String findFittingDatatypes(long n) {  List<String> fittingDatatypes = new ArrayList<>();  if (Byte.MIN\_VALUE <= n && n <= Byte.MAX\_VALUE) {  fittingDatatypes.add("byte");  }  if (Short.MIN\_VALUE <= n && n <= Short.MAX\_VALUE) {  fittingDatatypes.add("short");  }  if (Integer.MIN\_VALUE <= n && n <= Integer.MAX\_VALUE) {  fittingDatatypes.add("int");  }  if (Long.MIN\_VALUE <= n && n <= Long.MAX\_VALUE) {  fittingDatatypes.add("long");  }  if (fittingDatatypes.isEmpty()) {  return "N can't be fitted anywhere";  }  return "N can be fitted in:\n\* " + String.join("\n\* ", fittingDatatypes);  }  } |

Soal 2

Variables

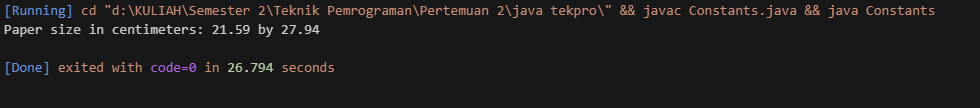
Perhatikan baris program dibawah ini:

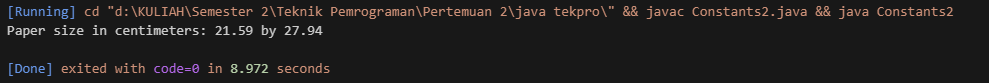
|  |
| --- |
| public class Constants {  public static void main(String[] args)  { final double CM\_PER\_INCH = 2.54;  double paperWidth = 8.5;  double paperHeight = 11;  System.out.println("Paper size in centimeters: " +  paperWidth \* CM\_PER\_INCH + " by " + paperHeight \*  CM\_PER\_INCH);  }  } |

|  |
| --- |
| 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?





1. Apa perbedaan penggunaan final double dengan public static final double?

Final double adalah deklarasi variabel dengan tipe data double yang nilai variabel tersebut tidak dapat diubah setelah inisialisasi pertama kali, jadi nilai variabel tersebut tetap konstan sepanjang program berjalan. Public satatic final double adalah kombinasi penggunaan kata kunci aksesibilitas ‘public’ menandakan bahwa variabel tersebut dapat diakses dari manapun dalam program, ‘static’ menunjukkan bahwa variabel tersebut milik kelas dan bukan objek individu, dan ‘final’ menjamin bahwa nilainya tidak dapat diubah setelah inisialisasi.

Soal 3

Operators

Perhatikan baris program dibawah 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);**

Nilai nx stelah menggunakan Math.round(x) adalah 93. Karena Math.round(x) akan mengembalikan nilai x yang sudah dibulatkan ke bilangan bulat terdekat, dan dalam hal ini nilai x adalah 92.98, yang jika dibulatkan akan menjadi 93.

1. Kenapa dibutuhkan cast (int) dalam penggunaan **Math.round(x) ?**

Casting (int0 digunakan dalam Math.round(x) karena Math.round(x) mengembalikan nilai bertipe data double, sedangkan nx adalah variabel bertipe data int. dengan menggunakan (int) sebelum Math.round(x) , nilai yang terkonversi yang dibulatkan tersebut menjadi tipe data int sehingga sesuai dengan tipe data variabel nx.

Soal 4

Operators (1)

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.

* Pertama, nilai dari d adalah 10.25.
* Kemudian, nilai d dikonversi menjadi float dengan melakukan cast (float) d, hasilnya adalah 10.25.
* Selanjutnya, nilai f (10.25) dikonversi menjadi long dengan melakukan cast (long) f, hasilnya adalah 10.
* Metode methodOne dipanggil dengan parameter f yang telah dikonversi menjadi long.
* Di dalam methodOne, nilai l (yang sekarang adalah 10) diubah menjadi int, yang tetap 10.
* Kemudian, nilai tersebut dikembalikan sebagai short.
* Akhirnya, nilai b adalah hasil dari nilai short tersebut yang dikonversi menjadi byte.
* Karena nilai 10 masih dalam rentang byte, maka nilai b tetap 10.
* Jadi, output nilai dari variabel b adalah 10.

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

Ketika nilai dari variabel d (10.25) dikonversi menjadi float, tidak ada perubahan signifikan, karena nilai float dapat menampung nilai double tanpa kehilangan presisi pada kasus ini. Namun, ketika nilai float tersebut dikonversi menjadi long, bagian desimal dibuang, sehingga dari 10.25 menjadi 10. Kemudian, nilai long tersebut dikonversi menjadi byte, yang menyebabkan pemotongan nilai jika nilai tersebut di luar rentang nilai yang dapat ditampung oleh byte. Dalam soal ini nilai 10 masih dalam rentang nilai yang dapat ditampung oleh byte, sehingga tidak ada perubahan signifikan dalam nilai dari d hingga nilai b. Jadi, meskipun ada beberapa konversi tipe data yang dilakukan, nilai dari variabel d hingga variabel b masih sama, yaitu 10.

Soal 5

Strings

This exercise is to test your understanding of Java Strings. A sample String declaration:

|  |
| --- |
| String myString = "Hello World!" |

The elements of a String are called characters. The number of characters in a String is called the length, and it can be retrieved with the String.length() method.

Given two strings of lowercase English letters, ***A*** and ***B***, perform the following operations:

1. Sum the lengths of ***A*** and ***B.***

2. Determine if ***A*** is lexicographically larger than ***B*** (i.e: does ***B*** come before ***A*** in the  
dictionary?)

3. Capitalize the first letter in ***A*** and ***B*** and print them on a single line, separated by a  
space.

**Input Format**

The first line contains a string ***A***. The second line contains another string ***B***. The strings are  
comprised of only lowercase English letters.

**Output Format**

There are three lines of output:

For the first line, sum the lengths of ***A*** and ***B***.

For the second line, write Yes if ***A*** is lexicographically greater than ***B*** otherwise print No instead.

For the third line, capitalize the first letter in both ***A*** and and ***B*** print them on a single line,  
separated by a space.

**Sample Input 0**

|  |
| --- |
| hello java |

**Sample Output 0**

|  |
| --- |
| 9N o Hello Java |

**Explanation 0**

String ***A*** is "hello" and ***B*** is "java".

***A*** has a length of 5, and ***B*** has a length of 4; the sum of their lengths is 9.

When sorted alphabetically/lexicographically, "hello" precedes "java"; therefore ***A***, is not  
greater than ***B*** and the answer is No.

When you capitalize the first letter of both ***A*** and ***B*** and then print them separated by a space,  
you get "Hello Java".

|  |
| --- |
| import java.util.Scanner;  public class Main {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  String A = scanner.nextLine();  String B = scanner.nextLine();  int sumLengths = A.length() + B.length();  System.out.println(sumLengths);  if (A.compareTo(B) > 0) {  System.out.println("Yes");  } else {  System.out.println("No");  }  String capitalizedA = A.substring(0, 1).toUpperCase() + A.substring(1);  String capitalizedB = B.substring(0, 1).toUpperCase() + B.substring(1);  System.out.println(capitalizedA + " " + capitalizedB);  }  } |