Pertemuan III
Input dan Output Sederhana

3.1. Konversi Tipe

Ketika membuat program, kita biasa memberikan nilai satu tipe dengan tipe lain. Jika dua tipe tersebut kompatibel, maka java akan melakukan konversi secara otomatis. Misalnya, jika kita memberikan nilai bertipe *int* ke variabel bertipe *long.*

Tidak semua tipe data adalah kompatibel, misalnya kita mengisi variabel bertipe *byte* dengan data bertipe *double.* Tetapi kita masih dapat melakukannya secara manual, tetapi kita harus memperhatikan konsekuensi yang mungkin terjadi, seperti hilangnya sebagian data.

3.1.1. Konversi Tipe Otomatis

Ketika satu variabel diisi dengan data bertipe lain, maka java akanmengkonversi tipe data tersebut secara otomatis bila memenuhi dua syarat berikut ini :

a. Kedua tipe adalah compatible

b. Tipe data tujuan memiliki ukuran lebih besar dibanding tipe sumber

Ketika kedua syarat diatas terpenuhi, maka konversi pelebaran ukuran akan dilakukan secara otomatis. Tipe data int lebih besar dari pada tipe data *byte,* ketika kita melakukan konversi dari data bertipe *byte* ke data bertipe *int,* kita tidak perlu melakukan konversi secara ekplisit.

Konversi pelebaran akan dilakukan pada bilangan bulat dan bilangan *floating-point* yang kompatibel antara satu dengan yang lain. Tipe data numerik tidak kompatibel dengan *char* atau *boolean.* Tipe *char* dan *boolean* juga tidak saling kompatibel.

Java juga secara otomatis melakukan konversi tipe ketika menyimpan literal bilangan bulat ke variabel bertipe *byte, short,* atau *long.*

3.1.2. *Casting*

Casting atau typecasting adalah proses konversi data dari tipe data tertentu ketipe data yang lain. Adakalanya kita perlu memaksa konversi dari tipe data tertentuke tipe data lain yang tidak kompatibel atau memiliki ukuran lebih kecil. Misalnyaketika fungsi mengirim tipe berbeda dari tipe yang diperlukan untuk operasi, maka tindakan casting diperlukan.

Tindakan casting dilakukan dengan menempatkan tipe data yang diharapkan didalam tanda kurung disebelah kiri nilai yang dikonversi. Sintaks casting adalah :

```
(TipeTarget) Nilai
```

TipeTarget adalah tipe data yang diinginkan, sedangkan Nilai adalah nilai yang akan dikonversi. Contoh penggunaannya dapat dilihat pada program dibawah ini :

ContohCasting.java

```
public class ContohCasting {
  public static void main(String args[]){
     int i1=17, i2=4;

     double d= (double) i1/i2;

     System.out.println(d);
     System.out.println((int) d);
   }
}
```

Keluaran dari program diatas adalah :

```
4.25
4
```

Jika proses *casting* pada pengisian variabel d, yaitu "(double)" dihilangkan maka keluarannya menjadi seperti :

```
4.0
4
```

Pengetahuan ukuran penyimpanan penting dalam menentukan hasil akhir *casting.* Tidak semua *asting* dapat menyelamatkan semua data sebelumnya secara utuh. Pada proses *casting long* menjadi *int,* berarti tipe *long* (64 bit) menjadi tipe *int* (32 bit). Jika ini dilakukan maka kompilator akan memotong 32 bit tipe data *long* menjadi 32 bit tipe data *int.* Jika nilai 32 bit bagian atas berisi informasi penting, maka informasi itu akan hilang.

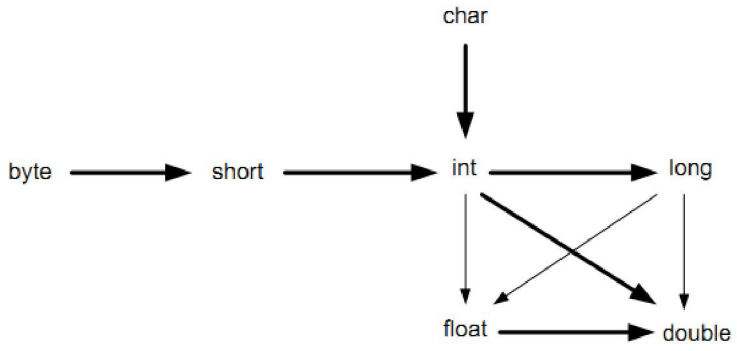
Informasi juga dapat hilang ketika kita melakukan *casting* antara tipe data yang berbeda, walupun memiliki ukuran penyimpanan yang sama. Misalnya kita melakukan *casting* bilangan *double* menjadi *long,* walaupun ukurannya sama 64 bit, hal ini akan menghilangkan nilai pecahannya.

Konversi yang mengakibatkan pemotongan disebut *truncation.* Pada tabel 3.1 berikut ini adalah aftar proses *casting* yang tidak mengakibatkan pemotongan dan hasilnya dijamin tidak ada yang ilang.

Tabel 3.1. *Casting* yang tidak mengalami *truncation*

Tipe Asal	Tipe Tujuan
byte	short, char, int, long, float, double
short	int, long, float, double
char	int, long, float, double
int	long, float, double
long	float, double
float	double

Alur konversi tipe-tipe data primitif dapat dilihat pada gambar 3.1 dibawah ini :



Gambar 3.1. Alur konversi tipe-tipe data primitif

Enam anak panah tebal menunjukkan konversi yang tidak menghilangkan informasi. Sedangkan tiga anak panah tipis menunjukkan konversi yang dapat menimbulkan kehilangan presisi. Contoh bilangan int 123456789 mempunyai banyak digit yang dapat direpresentasikan float, hasilnya mempunyai magnitudo yang benar namun menghilangkan presisi.

CastingInt2Float.java

```
public class CastingInt2Float {
  public static void main(String args[]){
     int N_int = 123456789;

     float N_float = N_int;

     System.out.println(N_int);
     System.out.println(N_float);
  }
}
```

Keluaran program diatas adalah :

```
123456789
1.23456792E8
```

Terlihat bahwa digit ke-8 dan ke-9 nilainya berubah dari 89 menjadi 92. Hal ini penting untuk diperhatikan dalam membuat program agar tidak terjadi kesalahan dalam operasi.

3.1.3. *Membaca* masukan dari keyboard

Kita telah mempelajari konsep dasar pemrograman pada Java dan menulis beberapa program Sederhana. Sekarang kita akan mencoba membuat program yang lebih interaktif dengan Menggunakan input dari keyboard. Pada bab ini, kita akan mempelajari dua cara memberikan input, yaitu menggunakan class BufferedReader dan melalui GUI (Graphical User Interface) dengan menggunakan class JOptionPane.

3.1.4. Menggunakan BufferedReader

Pada bagian ini, kita akan menggunakan class BufferedReader yang berada dipaket java.io untuk mendapatkan input dari keyboard.

Berikut ini adalah program untuk membaca input dari keyboard menggunakan class BufferedReader :

GetInputKeyboardBufferedReader.java

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

public class GetInputKeyboardBufferedReader{
   public static void main( String[] args ){
      BufferedReader dataIn = new BufferedReader(new
InputStreamReader(System.in));
      String nama = "";
      System.out.print("Ketik nama anda : ");

      try{
        nama = dataIn.readLine();
      }
      catch( IOException e ){
        System.out.println("Ada kesalahan !");
      }

      System.out.println();
      System.out.println("Hello " + nama +"\nLanjutkan belajarnya
pasti menjadi programmer Java !");
   }
}
```

Penjelasan dari program diatas adalah sebagai berikut:
Statement,

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
```

menjelaskan bahwa kita akan menggunakan class *BufferedReader*, *InputStreamReader* dan *IOException* yang berada di paket *java.io.* Java *Application Programming Interface* (API) berisi ratusan *class* yang sudah didefinisikan sebelumnya yang dapat digunakan untuk program kita. *Class-class* tersebut dikumpulkan di dalam paket-paket.
Paket berisi class yang mempunyai fungsi yang saling berhubungan. Seperti pada contoh di atas, paket *java.io* mengandung *class-class* yang memungkinkan program untuk melakukan *input* dan i*ntput* data. Pernyataan di atas juga dapat ditulis sebagai berikut :

```
import java.io.*
```

yang akan mengeluarkan semua *class* yang berada dalam paket, dan selanjutnya kita dapat menggunakan *class-class* tersebut dalam program kita.

Dua statement selanjutnya,

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

kita sudah mempelajari pada pelajaran sebelumnya. Pernyataan ini mendeklarasikan *class* bernama *GetInputKeyboardBufferedReader* dan kita. mendeklarasikan metode *main* yang akan diproses pertama kali ketika program dijalankan.
Dalam statement,

```
 BufferedReader dataIn = new BufferedReader(new
InputStreamReader(System.in));
```

kita mendeklarasikan sebuah variabel bernama *dataIn* dengan tipe *class BufferedReader*. *BufferedReader* adalah *class* yg disediakan oleh java untuk melakukan proses *input/output* oleh pengguna dari *keyboard* tanpa menggunakan fasilitas *Swing* atau AWT.

Sekarang, kita akan mendeklarasikan variabel nama bertipe *String* untuk menyimpan input dari pengguna melalui keyboard :

```
 String nama = "";
```

Variabel nama diinisialisasi sebagai *String* kosong "", Sebaiknya kita selalu menginisialisasi sebuah variabel setelah kita mendeklarasikannya.

Baris berikutnya adalah untuk menampilkan teks untuk berinteraksi dengan pengguna, yaitu untuk memerintahkan pengguna agar mengetikkan namanya.

```
System.out.print("Ketik nama anda : ");
```

Sekarang, blok di bawah ini merupakan blok *try-catch*,

```
try{
  nama = dataIn.readLine();
}
catch( IOException e ){
  System.out.println("Ada kesalahan !");
}
```

Untuk mengantisipasi jika terjadi kesalahan pada pernyataan:

```
nama = dataIn.readLine();
```

maka digunakan blok *try-catch* agar kesalahan yang terjadi tidak menggangu jalannya program. Kita akan membahas *try-catch* pada penanganan exception di bab selanjutnya. Sekarang kita cukup mencatat bahwa kita perlu menambahkan kode ini untuk menggunakan metode *readLine*() dari *BufferedReader* untuk mendapatkan masukan pengguna dari keyboard.

Selanjutnya kembali ke pernyataan :

```
nama = dataIn.readLine();
```

metode diatas memanggil *dataIn.readLine*(), untuk mendapatkan masukan dari pengguna melalui keyboard dan memberikan sebuah nilai *String*. Nilai ini akan disimpan ke dalam variabel nama, yang akan kita gunakan pada statement berikut ini :

```
System.out.println();
System.out.println("Hello " + nama +"\nLanjutkan belajarnya pasti
menjadi programmer Java !");
```

Baris pertama untuk membuat baris kosong, sehingga tulisan masukan dari pengguna memiliki jarak dengan tampilan berikutnya (Hello.....).

3.2.2. Menggunakan *JOptionPane*

Cara lain untuk mendapatkan masukan dari pengguna adalah dengan menggunakan *class JoptionPane* yang didapatkan dari paket *javax.swing*. *JoptionPane* memudahkan memunculkan kotak dialog standard yang memberikan kepada user sebuah nilai atau menginformasikan sesuatu.

Penggunaannya dapat dilihat pada program dibawah ini :

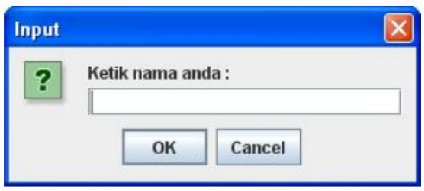
GetInputKeyboardJOptionPane.java

```
import javax.swing.JOptionPane;

public class GetInputKeyboardJOptionPane{
   public static void main( String[] args ){
      String nama = "";

      nama = JOptionPane.showInputDialog("Ketik nama anda : ");
      String msg = "Hello " + nama +"\nLanjutkan belajarnya pasti
menjadi programmer Java !";
      JOptionPane.showMessageDialog(null, msg);
   }
}
```

Tampilan dari program diatas adalah :



Gambar 3.2. Tampilan *input* menggunakan *JoptionPane*

Penjelasan program diatas adalah sebagai berikut :
Statement pertama :

```
import javax.swing.JOptionPane;
```

Menjelaskan bahwa kita mengimpor *class JoptionPane* dari paket *javax.swing*, atau bisa ditulis seperti pernyataan berikut ini :

```
import javax.swing.*;
```

Tiga baris pernyataan berikutnya sudah dijelaskan pada bagian input keyboard menggunakan BufferedReader. Pernyataan berikut ini :

```
nama = JOptionPane.showInputDialog("Ketik nama anda : ");
```

digunakan untuk membuat sebuah kotak dialog *JOptionPane*, yang akan menampilkan dialog dengan sebuah pesan, sebuah *textfield* dan tombol Ok seperti pada gambar 3.2. Hasil dari dialog tersebut adalah *String* dan disimpan ke dalam variabel nama.

Pernyataan selanjutnya adalah mendefinisikan variabel msg bertipe String dan diisi dengan serangkaian teks :

```
String msg = "Hello " + nama +"\nLanjutkan belajarnya pasti menjadi
programmer Java !";
```

Baris selanjutnya adalah menampilkan sebuah dialog yang berisi sebuah pesan dan tombol Ok :

```
JOptionPane.showMessageDialog(null, msg);
```

3.3. Format keluaran

Java memisahkan komponen untuk menampilkan keluaran dengan komponen untuk melakukan format keluaran. Keuntungan pemisahan ini antara lain dapat memberikan format keluaran yang lebih banyak, melebihi yang disediakan oleh C++.

Paket *java.text* menyediakan beragam format tampilan yang biasa ditemui di kehidupan sehari-hari. Terdapat tiga metode utama, yaitu format terhadap angka, mata uang dan persentase.

Untuk memperoleh *formatter* (obyek pelaku format) maka kita dapat memanggil :

- getNumberInstance()
- getCurrencyInstance()
- getPercentInstance()

Metode mengirim obyek bertipe *NumberFormat*. Kita dapat menggunakan obyek itu untuk melakukan format satu angka atau lebih. Kita kemudian menerapkan metode format ke obyek itu untuk melakukan format yang diinginkan, sehingga memperoleh string yang berisi angka yang telah terformat.

Metode di *NumberFormat* untuk mengkonfigurasi format angka dapat dilihat pada tabel 2.2 dibawah ini :

Tabel 2.2. Metode di *NumberFormat*

No	Metode	Deskripsi
1	`static NumberFormat getCurrencyInstance()`	Mengirim obyek NumberFormat yang bertugas mengkonversi ke string yang menunjukkan mata uang menggunakan konvensi lokal
2	`static NumberFormat getNumberInstance()`	Mengirim obyek NumberFormat yang bertugas mengkonversi ke string yang menunjukkan angka menggunakan konvensi lokal
3	`static NumberFormat getPercentInstance()`	Mengirim obyek NumberFormat yang bertugas mengkonversi ke string yang menunjukkan persentase menggunakan konvensi lokal
4	`void setMaximumFractionDigits(int digits)`	Mengeset jumlah angka maksimum setelah koma
5	`void setMaximumIntegerDigits(int digits)`	Mengeset jumlah angka maksimum sebelum koma
6	`void setMinimumFractionDigits(int digits)`	Mengeset jumlah angka minimum setelah koma
7	`void setMinimumFractionDigits(int digits)`	Mengeset jumlah angka minimum sebelum koma

Untuk menggunakan metode format, kita harus meng-*import java.text.** karena *NumberFormat* terdapat di paket *java.text*. Contoh aplikasi metode format dapat dilihat pada program dibawah ini :

FormatAngka.java

```
import java.text.*;

public class FormatAngka{
   public static void main(String args[]){
      double Angka=83243463.342245;
      double AngkaPecahan=0.902235643;

      NumberFormat NumberFormatter = NumberFormat.getNumberInstance();
      NumberFormat CurrFormatter = NumberFormat.getCurrencyInstance();
      NumberFormat PercentFormatter =
NumberFormat.getPercentInstance();

      String NumberStr = NumberFormatter.format(Angka);
      String CurrStr = CurrFormatter.format(Angka);
      String PercentStr = PercentFormatter.format(AngkaPecahan);

      System.out.println("double Angka = "+ Angka +" berformat number
: "+NumberStr);
      System.out.println("double Angka = "+ Angka +" berformat
currency : "+CurrStr);
      System.out.println("double Angka = "+ AngkaPecahan +" berformat
percent : "+PercentStr);
      System.out.println();
      System.out.println();

      NumberFormatter.setMaximumIntegerDigits(10);
      CurrFormatter.setMaximumIntegerDigits(10);
      PercentFormatter.setMaximumIntegerDigits(10);

      NumberFormatter.setMinimumIntegerDigits(1);
      CurrFormatter.setMinimumIntegerDigits(1);
      PercentFormatter.setMinimumIntegerDigits(1);

      NumberFormatter.setMaximumFractionDigits(5);
      CurrFormatter.setMaximumFractionDigits(2);
      PercentFormatter.setMaximumFractionDigits(4);

      NumberFormatter.setMinimumFractionDigits(2);
      CurrFormatter.setMinimumFractionDigits(2);
      PercentFormatter.setMinimumFractionDigits(6);

      System.out.println("double Angka = "+ Angka +" berformat number
: "+NumberFormatter.format(Angka));
      System.out.println("double Angka = "+ Angka +" berformat
currency : "+CurrFormatter.format(Angka));
      System.out.println("double Angka = "+ AngkaPecahan +" berformat
percent : "+PercentFormatter.format(AngkaPecahan));
   }
}
```

Keluaran dari program diatas adalah sebagai berikut :

```
double Angka = 8.3243463342245E7 berformat number : 83.243.463,342
double Angka = 8.3243463342245E7 berformat currency : Rp83.243.463,34
double Angka = 0.902235643 berformat percent : 90%


double Angka = 8.3243463342245E7 berformat number : 83.243.463,34224
double Angka = 8.3243463342245E7 berformat currency : Rp83.243.463,34
double Angka = 0.902235643 berformat percent : 90,223564%
```

Kita dapat memperoleh format angka yang cocok untuk lokal-lokal berbeda. Misalnya kita ingin membuat format sesuai lokal jerman, maka kita dapat memanggil metode :

- `getNumberInstance(Locale.GERMANY)`
- `getCurrencyInstance(Locale.GERMANY)`
- `getPercentInstance(Locale.GERMANY)`

Karena metode *Locale* berada di paket *java.util*, maka harus meng-*import* *java.util.** untuk dapat menggunakannya. Contoh penggunaan di program dapat dilihat pada program dibawah ini :

FormatAngkaLokal.java

```
import java.text.*;
import java.util.*;

public class FormatAngkaLokal{
  public static void main(String args[]){
     double Angka=83243463.342245;
     double AngkaPecahan=0.902235643;

     NumberFormat NumberFormatterGERMANY =
NumberFormat.getNumberInstance(Locale.GERMANY);
     NumberFormat CurrFormatterGERMANY =
NumberFormat.getCurrencyInstance(Locale.GERMANY);
     NumberFormat PercentFormatterGERMANY =
NumberFormat.getPercentInstance(Locale.GERMANY);

     String NumberStrGERMANY = NumberFormatterGERMANY.format(Angka);
     String CurrStrGERMANY = CurrFormatterGERMANY.format(Angka);
     String PercentStrGERMANY =
PercentFormatterGERMANY.format(AngkaPecahan);
```

```
      System.out.println("double Angka = "+ Angka +" berformat number
: "+NumberStrGERMANY);
      System.out.println("double Angka = "+ Angka +" berformat
currency : "+CurrStrGERMANY);
      System.out.println("double Angka = "+ AngkaPecahan +" berformat
percent : "+PercentStrGERMANY);
      System.out.println();
      System.out.println();

      NumberFormat NumberFormatterUS =
NumberFormat.getNumberInstance(Locale.US);
      NumberFormat CurrFormatterUS =
NumberFormat.getCurrencyInstance(Locale.US);
      NumberFormat PercentFormatterUS =
NumberFormat.getPercentInstance(Locale.US);

      String NumberStrUS = NumberFormatterUS.format(Angka);
      String CurrStrUS = CurrFormatterUS.format(Angka);
      String PercentStrUS = PercentFormatterUS.format(AngkaPecahan);

      System.out.println("double Angka = "+ Angka +" berformat number
: "+NumberStrUS);

      System.out.println("double Angka = "+ Angka +" berformat number
: "+NumberStrUS);
      System.out.println("double Angka = "+ Angka +" berformat
currency : "+CurrStrUS);
      System.out.println("double Angka = "+ AngkaPecahan +" berformat
percent : "+PercentStrUS);
    }
  }
```

Keluaran dari program diatas adalah sebagai berikut :

```
double Angka = 8.3243463342245E7 berformat number : 83.243.463,342
double Angka = 8.3243463342245E7 berformat currency : 83.243.463,34 Ç
double Angka = 0.902235643 berformat percent : 90%


double Angka = 8.3243463342245E7 berformat number : 83,243,463.342
double Angka = 8.3243463342245E7 berformat currency : $83,243,463.34
double Angka = 0.902235643 berformat percent : 90%
```

Referensi:

1. Hariyanto, Bambang, (2007), *Esensi-esensi Bahasa Pemrograman Java*, Edisi 2, Informatika Bandung, November 2007.
2. Utomo, Eko Priyo, (2009), *Panduan Mudah Mengenal Bahasa Java*, Yrama Widya, Juni 2009.
3. Tim Pengembang JENI, JENI 1-6, Depdiknas, 2007
4. SatriaWahono, Romi, *Java Advanced*, ilmukomputer.com