MODUL 1
DASAR PEMROGRAMAN JAVA

A. TUJUAN

- Praktikan mampu memahami tentang dasar-dasar pemrograman java.
- Praktikan mampu memahami tentang deklarasi variabel, operator, kondisional & pilihan, serta string.
- Praktikan dapat mengimplementasikannya dalam program sederhana dengan menggunakan NetBeans.

B. PERANGKAT

- NetBeans IDE 7.2

C. DASAR TEORI

1. Deklarasi Variabel

Bahasa pemrograman pada umumnya mengenal adanya variabel yang digunakan untuk menyimpan nilai atau data. Java dikenal dengan bahasa pemrograman yang bersifat *strongly typed* yang artinya diharuskan mendeklarasikan tipe data dari semua variabel dan apabila lupa atau salah mengikuti aturan pendeklarasian variabel maka akan mendapat *error* pada saat proses kompilasi.

1.1 Tipe data

Java memiliki dua jenis tipe data yang dikategorikan menjadi dua yaitu tipe data primitif dan tipe data referensi.

1.1.1 Tipe Data Primitif

Macam tipe data primitif diantaranya :

- **Integer (Bilangan bulat)**

  Integer merupakan tipe data numerik yang digunakan untuk mendefinisikan bilangan bulat. Tipe data numeric yang termasuk integer diantaranya :

Tabel 1.1 Tipe Data Primitif Integer

Tipe	Deskripsi
Byte	-128 s/d +127 menempati 8 bits di memori
Short	-32768 s/d +32767 menempati 16 bits di memori
Int	-2147483648 s/d +2147483647 menempati 32 bits di memori
Long	-9223372036854775808 s/d +9223372036854775807 menempati 64 bits di memori

- **Floating Point (Bilangan pecahan)**

  Floating point digunakan untuk menangani bilangan decimal atau perhitungan yang lebih detail dibanding integer.

Tabel 1.2 Tipe Data Primitif Floating Point

Tipe	Deskripsi
Float	$-3.4x10^{8}$ s/d $+3.4x10^{8}$
Double	$-1.7x10^{308}$ s/d $+1.7x10^{308}$

- **Char**

  Char adalah karakter tunggal yang pendefinisiannya di awal dan akhir menggunakan tanda petik tunggal (‘). Tipe char mengikuti aturan Unicode, sehingga bisa dapat menggunakan kode untuk kemudian diikuti bilangan dari 0 sampai 65535, tetapi yang biasa digunakan adalah bilangan heksadesimal dari 0000 sampai FFFF.

- **Boolean**

  Tipe data Boolean terdiri dari dua nilai saja, yaitu *true* dan *false*. Boolean sangat penting untuk mengevaluasi suatu kondisi.

1.1.2 Tipe Data Referensi

Kelebihan pemrograman dengan orientasi objek adalah dapat mendefinisikan tipe data baru yang merupakan objek dari *class* tertentu. Tipe data ini digunakan untuk mereferensikan objek atau class tertentu, seperti String.

- **Variabel**

  Variabel merupakan *container* yang digunakan untuk menyimpan suatu nilai pada sebuah program dengan tipe tertentu. Untuk mendefinisikan variabel, suatu identifier dapat digunakan untuk menamai variabel tersebut.

- **Identifier**

  Identifier adalah kumpulan karakter yang dapat digunakan untuk menamai variabel, method, class, interface, dan package. Dalam pemrograman Java identifier bisa disebut sah apabila diawali dengan :

  - Huruf /abjad
  - Karakter Mata Uang
  - Underscore(_)

  Identifier dapat terdiri dari :

  - Huruf / abjad
  - Angka
  - Underscore (_)

  Identifier tidak boleh mengandung @, spasi atau diawali dengan angka serta tidak boleh menggunakan *keyword* yang telah digunakan di pemrograman java. Selain karakter, Unicode juga dapat digunakan sebagai identifier.

1.2 Mendeklarasikan Variabel

Sintaks dasar :

[tipe data] [nama variabel]

Menuliskan tipe data dari variabel, contoh :

```
int bilangan;
char karakter;
float bildesimal;
boolean status;
```

{Setelah dideklarasikan sesuai dengan tipe data, selanjutnya memberi nilai variabel tersebut dengan tanda =}

```
Bilangan = 20;
Karakter = 'k';
Bildesimal = 22.2f;
Status = true;
```

{melakukan deklarasi tipe data dan memberi nilai variabel dalam satu baris}

```
int bilangan = 20;
char karakter = 'k';
boolean status = true;
```

{membuat variabel menjadi konstanta sehingga tidak bisa diubah lagu nilainya dengan menambahkan keyword sebelum tipe data}

```
Final int a = 10;
Final float pajak = 15.5;
```

{membuat agar konstanta dapat diakses oleh kelas lain tanpa harus membuat objek terlebih dulu dilakukan penambahan modifier public dan keyword static}

```
Public static final a = 10;
```

2. Operator

Operator adalah simbol khusus yang menyajikan operasi khusus pada satu, dua, atau tiga operand dan kemudian mengembalikan hasilnya.

Jenis operator antara lain :

2.1 Operator Aritmatika

Operator ini digunakan pada operasi-operasi aritmatika seperti penjumlahan, pengurangan, pembagian dan lain-lain.

Tabel 1.3 Operator Aritmatika

Operator	Keterangan
+	Penjumlahan
-	Pengurangan
*	Perkalian
/	Pembagian
%	Modulus (sisa bagi)
++	Increment (menaikkan nilai dengan 1)
--	Decrement (menurunkan nilai dengan 1)

2.2 Operator Relasional

Untuk membandingkan 2 nilai (variabel) atau lebih digunakan operator relasional, dimana operator ini akan mengembalikan atau menghasilkan nilai *True* atau *False*.

Tabel 1.4 Operator Relasional

Operator	Keterangan
==	Sama dengan
!=	Tidak sama dengan
>	Lebih besar
<	Lebih kecil
>=	Lebih besar atau sama dengan
<=	Lebih kecil atau sama dengan

2.3 Operator Kondisional

Operator ini menghasilkan nilai yang sama dengan operator relasional, hanya saja penggunaannya lebih pada operasi – operasi Boolean.

Tabel 1.5 Operator Kondisional

Operator	Keterangan
&&	Operasi AND
||	Operasi OR
^	Operasi XOR
!	Operasi NOT (Negasi)

Tabel 1.6 Contoh Operasi Kondisional

A	B	A&&B	A||B	A^B	!A
True	True	True	True	False	False
TRUE	False	False	True	True	False
False	True	False	True	True	True
False	False	False	False	False	True

2.4 Operator Shift dan Bitwise

Kedua operator ini digunakan untuk memanipulasi nilai dari bitnya, sehingga diperoleh nilai yang lain.

Tabel 1.7 Operator Shift dan Bitwise

Operator	Keterangan
&	Operasi bitwise AND
|	Operasi bitwise OR
^	Operasi bitwise XOR
~	Operasi bitwise NOT
>>	Operasi shift right (geser ke kanan sebanyak n bit)
>>>	Operasi shift right zero fill
<<	Operasi shift left (geser ke kiri sebanyak n bit)

Tabel 1.8 Contoh Operasi Bitwise

A	B	A&B	A|B	A^B	~A
1	1	1	1	0	0
1	0	0	1	1	0
0	1	0	1	1	1
0	0	0	0	0	1

2.5 Operator Assignment

Operator assignment dalam java digunakan untuk memberikan suatu nilai ke sebuah variabel. Operator assignment hanya berupa '=', namun selain itu dalam Java beberapa shortcut assignment operator yang penting.

Tabel 1.9 Operator Assignment

Operator	Contoh	Ekivalen dengan
+=	b+=a	b=b+a
-=	b-=a	b=b-a
*=	b*=a	b=b*a
/=	b/=a	b=b/a
%=	b%=a	b=b%a
&=	a&=b	a=a&b
|=	a|=b	a=a|b
^=	a^=b	a=a^b
<<=	a<<=b	a=a<<b
>>=	a>>=b	a=a>>b
>>>=	a>>>=b	a=a>>>b

Contoh 1:

```
public class contoh1a {
public static void main (String[]args){
int x, y, c;
x=4&6;
y=5>>>2;
c=(2+4)*6;
System.out.println(y);
System.out.println(c);
System.out.println(x);
```

```
public class contoh1b {
public static void main (String[]args) {
int a = 20;
int b = 10;
System.out.println("Hasil dari a%b =" +(a%b));
System.out.println("Hasil dari a*b =" +(a*b));
System.out.println("Hasil dari a ditambah" +(a++));
}}
```

Contoh 2:

```
public class Konversi{
public static void main(String args[]){
float m, cm, inci;
m = 30;
cm = m * 100;
inci = m * 100 / 2.54f;
System.out.println("Ukuran dalam CM = " + cm);
System.out.println("Ukuran dalam Inci = " + inci);}}
```

3. Kondisional dan Pilihan

3.1 IF

Statement if memungkinkan sebuah program untuk dapat memilih beberapa operasi untuk dieksekusi, berdasarkan beberapa pilihan. Terdapat tiga jenis statement If diantaranya :

- **If**

  Bentuk If adalah yang paling sederhana, mengandung suatu pernyataan tunggal yang dieksekusi jika ekspresi bersyarat adalah benar.

  Sintaks dasar:

```
If (ekspresi_kondisional) {
statement1;
statement2;
…
}
```

- **If, else**

  Untuk melakukan beberapa operasi yang berbeda jika salah satu ekspresi kondisional bernilai salah, maka digunakan statement else. Bentuk if-else memungkinkan dua alternatif operasi pemrosesan.

  Sintaks dasar:

```
If (ekspresi_kondisional) {
   statement1;
   statement2;
…
}else {
   statement1;
   statement2;
…
}
```

- **If, else if, else**

  Bentuk if, else if, else memungkinkan untuk tiga atau lebih alternative pemrosesan.

  Sintaks dasar:

```
If (ekspresi_kondisional) {
   statement1;
   statement2;
…
}else if (ekspresi_kondisional){
   statement1;
   statement2;
…
} else {
   statement1;
   statement2;
…
}
```

Contoh :

1. **If**

```
public class IfSatuPilihan{
public static void main(String args[]){
int bil;
bil=0;
if (bil==0)
System.out.println("Bilangan Nol");
}}
```

2. **If, else**

```
import java.util.Scanner;
public class IfDuaPilihan{
public static void main(String args[]){
      Scanner masuk = new Scanner(System.in);
      int bil;
```

```
        System.out.print("Masukkan bilangan : ");
        bil=masuk.nextInt();
        if (bil==0)
        System.out.println("Bilangan Nol");
        else
        System.out.println("Bilangan Bukan Nol");}}
```

3. If, else if, else

```
import java.util.Scanner
public class Buah{
public static void main(String args[]){Scanner masuk =
new Scanner(System.in);
int pil;
System.out.print("Masukkan pilihan : ");
pil = masuk.nextInt();
if (pil==1)
System.out.println("Buah Pepaya");
else if(pil==2)
System.out.println("Buah Stroberi");
else if(pil==3)
System.out.println("Buah Apel");
else
System.out.println("Bukan Buah deh");}}
```

3.2 Switch

Switch adalah pernyataan yang digunakan untuk menjalankann salah satu pernyataan dari beberapa kemungkinan statement untuk dieksekusi, berdasarkan nilai dari sebuah ungkapan dan nilai penyeleksi. Setiap ungkapan diungkapkan dengan sebuah nilai integral konstan, seperti sebuah nilai dengan tipe byte, short, int atau char.

Sintaks dasar :

```
switch (ekspresi){
      case value1:
                  statement1;
                  statement2;
                  break;
  case value2:
                  statement1;
                  statement2;
                  break;
            …
            [default:]
                  statement1;
                  statement2;
      }
```

Keterangan

Case : menandai posisi kode dimana eksekusi dilaksanakan.

Value1, dst : konstanta integer atau karakter ataupun ekspresi yang mengevaluasi keduanya.

Default : berfungsi sama seperti else pada statement if.

Break : dapat menghentikan perulangan walaupun kondisi untuk berhenti belum terpenuhi.

Continue :dengan statement ini kita bisa melewatkan operasi yang dilakukan dalam iterasi sesuai dengan kondisi tertentu.

Contoh :

```
import java.util.Scanner;
public class CaseJurusan{
    public static void main(String args[]){
    Scanner masuk = new Scanner(System.in);
          int pil;
          System.out.print("Masukkan pilihan :S1 TT ");
          pil = masuk.nextInt();
          switch (pil) {
           case 1:System.out.println("S1 TE");
            break;
            case  2:System.out.println("S1 SK");
            break;
           case  3:System.out.println("D3 TT");
            break;
           case  4:System.out.println("D3 IF");
            break;
           case 5:System.out.println("S1 TI");
            break;
          default:
            System.out.println("Input salah!");
            break;
            }
          }
          }
```