Komputer Grafik I - Pertemuan ke-2: Tutorial Pemrograman dengan Bahasa Java

1 Tentang Java

1.1 Bahasa Pemrograman Java

Java merupakan bahasa Pemrograman sekaligus Platform yang dikembangkan oleh James Gosling ketika masih bergabung degan Sun Microsystem (Sekarang Oracle) dan dirilis tahun 1995. Bahasa pemrograman Java banyak mengadopsi sintaks yang terdapat pada bahasa C/C++, sehingga penulisannya mirip sekali dengan bahasa C/C++. Hal ini sangat memudahkan programer yang mempunyai latar belakang bahasa C/C++ untuk mempelajari Java.

Seperti dijelaskan pada [1]Gambar 1, dalam Java, program pertama ditulis dengan menggunakan text editor sembarang, lalu disimpan ke file dengan ekstensi `.java`. Perhatikan, nama file harus sama dengan nama *class* pada program tersebut. Kode sumber yang sudah disimpan kemudian dikompile dengan `javac` menjadi `.class` file. Sebuah `.class` file berisi *bytecodes* yang merupakan bahasa mesin yang dapat dipahami oleh Java Virtual Machine (Java VM). Java VM (`java`) inilah yang bertugas menterjemahkan *bytecodes* tersebut menjadi bahasa mesin yang dapat dijalankan di berbagai sistem operasi.

Figure 1: Proses Pembuatan dan Eksekusi Program dengan Java

Sebuah program Java, terdiri dari kumpulan *class* dan sebuah *method* `main`. [1]Gambar 2 menunjukkan program sederhana pada bahasa Java dan bagaimana program tersebut dijalankan diberbagai sistem operasi yang berbeda.

1.2 Platform Java

Platform adalah lingkungan perangkat keras atau perangkat lunak dimana sebuah aplikasi dijalankan. *Platform* Java terdiri dari ([1]Gambar 3):

1. *Java Virtual Machine (Java VM)*
2. *Java Application Programming Interface (API)*

[1]http://download.oracle.com/javase/tutorial/getStarted/intro/definition.html

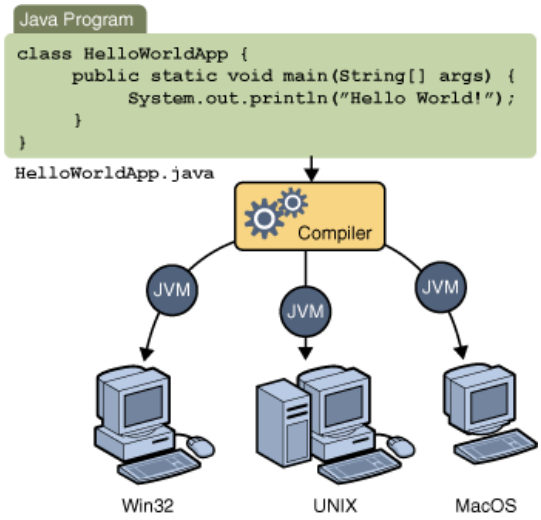

Figure 2: Aplikasi ”Hello World” pada Java

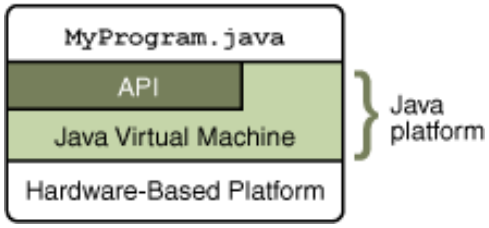

Figure 3: Java VM dan API

2 Dasar Pemrograman Java

2.1 Variabel

Secara garis besar, sebuah aplikasi komputer menerima ”suatu” masukan kemudian memprosesnya untuk menghasilkan ”suatu” keluaran. ”Sesuatu” tersebut umumnya disebut data dan agar dapat diproses dari program, harus disimpan di memori lalu diacu dengan menggunakan **variabel**, contohnya pada List 1.

Listing 1: Contoh variabel

```
1     int X = 10;              //bilangan bulat
2     double ipk;
3     char grade = 'A';        //huruf
4     String nama = "Sm*sh";   //variabel tipe string (kumpulan huruf)
5     ipk = 3.0;               //baru di set nilainya
6     float bunga = 1.0f;      //32-bit floating point
```

Dalam bahasa Java, ada beberapa jenis variabel, yaitu:

1. *Instance Variables (Non-static fields)*: Variabel untuk menyimpan ”state” dari suatu objek, unik untuk setiap objek.
2. *Class Variables (Static fields)*: Variabel yang di-share dalam sebuah *class* dan hanya ada satu.
3. *Local Variables*: Variabel yang digunakan untuk menyimpan nilai sementara dalam sebuah *method*.
4. *Parameters*: Variable yang digunakan untuk memberikan nilai ke *method*.

2.1.1 Penamaan Variabel

List 1 memuat contoh deklarasi dan penggunaan variabel. Sebuah variabel mempunyai **nama** dan **tipe** dari data yang bisa disimpan pada variabel tersebut. Nama dari suatu variabel adalah *case-sensitif* (dibedakan antara huruf besar dan huruf kecil). Kemudian, penamaan variabel harus mengikuti kaidah-kaidah seperti dibawah ini:

1. Kombinasi antara huruf (dalam Unicode), digit (0-9), tanda $, dan garis bawah (_).
2. Diawali dengan huruf, tanda $ atau garis bawah(_)
3. Tidak boleh sama dengan *keywords*

Keywords adalah kata-kata yang mempunyai makna khusus/tertentu dalam struktur bahasa pemrograman. Berikut adalah *keywords* pada bahasa Java.

Listing 2: Keywords

```
abstract    continue    for         new         switch
assert***   default     goto*       package     synchronized
boolean     do          if          private     this
break       double      implements  protected   throw
byte        else        import      public      throws
case        enum****    instanceof  return      transient
catch       extends     int         short       try
char        final       interface   static      void
class       finally     long        strictfp**  volatile
const*      float       native      super       while
```

* tidak digunakan, ** ditambahkan pada versi 1.2, *** ditambahkan pada versi 1.4, **** ditambahkan pada versi 5.0

2.1.2 Tipe Data Primitif

Seperti dijelaskan sebelumnya, sebuah variabel mempunya tipe tertentu. Berikut adalah tipe data primitif, yaitu tipe data yang sudah disediakan pada kompiler (bukan merupakan pustaka / *library*).

Table 1: Tipe data primitif pada bahasa Java

Nama	Lebar Data	Rentang Nilai
byte	8-bit	-128 s/d 127
short	16-bit	-32,768 s/d 32,767
int	32-bit	-2,147,483,648 s/d 2,147,483,647
long	64-bit	-9,223,372,036,854,775,808 s/d 9,223,372,036,854,775,807
float	32-bit	7 angka dibelakang koma
double	64-bit	14-15 angka dibelakang koma
boolean	-	true, false
char	16-bit Unicode	'\u0000' s/d '\uffff'

2.1.3 Array

Array adalah sebuah objek kontainer yang terdiri dari beberapa elemen dengan tipe yang sama. Panjang atau jumlah elemen dari sebuah array adalah konstan dan ditentukan pada saat objek tersebut dialokasikan. Gambar 4 menunjukkan sebuah *array* dengan jumlah elemen 10, sedangkan List 3 menunjukkan contoh kode *array* dalam bahasa Java. Setiap elemen dari array dapat diakses dengan **indek** (mulai dari 0).

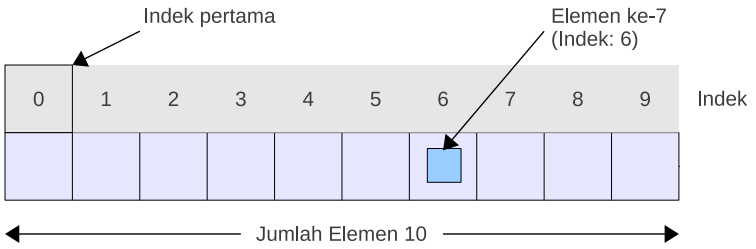

Figure 4: Array

Listing 3: Contoh kode Array

```
/*
 * Diadopsi dari:
 *   http://download.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html
 */
class ArrayDemo {
    public static void main(String[] args) {
        int[] anArray;              // Deklarasi variabel array tipe integer

        anArray = new int[10];      // Alokasikan memori untuk 10 buah elemen

        anArray[0] = 100;           // isi nilai elemen pertama
        anArray[1] = 200;           // isi nilai elemen kedua
        anArray[2] = 300;           // dst
        anArray[3] = 400;
        anArray[4] = 500;
        anArray[5] = 600;
        anArray[6] = 700;
        anArray[7] = 800;
        anArray[8] = 900;
        anArray[9] = 1000;

        // Mengakses elemen-elemen array
        System.out.println("Elemen pada indek 0: " + anArray[0]);
        System.out.println("Elemen pada indek 1: " + anArray[1]);
        System.out.println("Elemen pada indek 2: " + anArray[2]);
        System.out.println("Elemen pada indek 3: " + anArray[3]);
        System.out.println("Elemen pada indek 4: " + anArray[4]);
        System.out.println("Elemen pada indek 5: " + anArray[5]);
        System.out.println("Elemen pada indek 6: " + anArray[6]);
        System.out.println("Elemen pada indek 7: " + anArray[7]);
        System.out.println("Elemen pada indek 8: " + anArray[8]);
        System.out.println("Elemen pada indek 9: " + anArray[9]);
    }
}
```

Selain array 1-D, dalam java dimungkinkan untuk mendefinisikan array multidimensi. Sebenarnya, array multidimensi hanyalah array yang elemennya adalah array juga. Pada List 4 ditampilkan contoh array multidimensi yang terdiri dari 2 (dua) elemen (baris), dimana elemen pertama terdiri dari 2 (dua) elemen, sedangkan elemen kedua terdiri dari 3 (tiga) elemen. Untuk mengkopi elemen-elemen array ke array yang lain, disediakan *method*:

```
public static void arraycopy(Object src, int srcPos,
    Object dest, int destPos, int length)
```

Listing 4: Contoh kode Array Multidimensi

```
/*
 * Contoh array multidimensi
 */
class MultiDimArrayDemo {
    public static void main(String[] args) {
        String[][] names = {
            {"Soal ", "Jawaban "},
            {"Pemrograman Java", "Algoritma & Pemrograman", "Komputer Grafik"}
        };
        System.out.println(names[0][0] + names[1][0]); //Soal Pemrograman Java
        System.out.println(names[0][1] + names[1][2]); //Jawaban Komputer Grafik
    }
}
```

2.2 Operator

Dalam bahasa pemrograman Java disediakan operator untuk melakukan operasi aritmatika, logika, bit, komparasi dan assignment. Pada intinya, persamaan (expresi) akan dievaluasi dari kiri ke kanan dengan memperhatikan skala prioritas seperti pada Tabel 2.

Table 2: Operator dan Urutan Prioritas

Operator	Prioritas
postfix	$expr++, expr--$
unary	$++expr, --expr, +expr, -expr, \sim, !$
multiplicative	$*, /, \%$
additive	$+, -$
shift	$<<, >>, >>>$
relational	$<, <=, >, >=, instanceof$
equality	$==, !=$
bitwise AND	&
bitwise XOR	^
bitwise OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	=, +=, -=, *=, /=, %=, &=, ^=, <<=, >>=, >>>=

Yang perlu mendapat perhatian disini terkait operator adalah pembagian. Pembagian antar bilangan bulat (`int`) dan bilangan bulat (`int`), hasilnya adalah bilangan bulat (`int`). Dengan kata lain `1/2` hasilnya adalah `0` dan bukan `0.5`, sedangkan `1.0/2` hasilnya adalah `0.5`, karena `1.0` adalah `double`, sehingga hasilnya dalam bentuk `double`. Kemudian, bilangan sembarang dibagi dengan `0`, hasilnya adalah `NaN` (bukan bilangan).

2.3 Ekspresi, Pernyataan dan Blok

Variabel dan operator adalah unit terkecil dari pemrograman. Kombinasi antara variabel dan operator akan menghasilkan ekspresi (*expression*). `1+x`, `x+y/3`, `nilai>=80`, dan sebagainya adalah contoh ekspresi.

Komponen selanjutnya adalah pernyataan (*statements*). Pernyataan hampir sama dengan kalimat dalam bahasa manusia. Pernyataan biasanya diakhiri dengan tanda titik koma (;). Sekumpulan pernyataan akan membentuk blok (*block*). Sebuah blok ditandai dengan kurung kurawal buka { dan kurung kurawal tutup }.

Listing 5: Ekspresi, pernyataan dan blok

```
1     int x = 100;              // pernyataan
2     int y = 200;              // pernyataan
3     int z = x + y;            // pernyataan, sedangkan x + y adalah ekspresi
4
5     int sum = 0, idx = 0;
6     while (idx < 10) {        // awal blok, idx < 10 adalah ekspresi
7         sum += idx;
8         idx ++;
9     }                         // akhir blok
```

2.4 *Control-flow*

Pada dasarnya pernyataan-pernyataan dalam program dieksekusi dari atas ke bawah secara berurutan (**runtunan**). Namun ada kalanya diperlukan pengambilan keputusan , pengulangan atau percabangan dalam sebuah program. Hal ini memungkinkan program kita mengeksekusi ataupun mengulang sebuah blok pernyataan berdasarkan kriteria atau kondisi tertentu. Dalam pokok bahasan ini akan dibahas mengenai pengambilan keputusan: `if-then`, `if-then-else`, `switch` dan pengulangan: `while, for, do-while` serta percabangan: `continue, break, return`.

2.4.1 if-then

Pengambilan keputusan yang pertama menggunakan struktur `if-then`. Dalam struktur pengambilan keputusan seperti ini, jika ekpresi bernilai benar, maka blok pernyataan yang ada dalam `if` akan dieksekusi, jika nilainya salah, akan diabaikan.

Listing 6: Contoh if-then

```
1     int absolute(int nilai) {
2         if (nilai < 0) {        // blok pernyataan dieksekusi jika nilai minus
3             return -nilai;
4         }
5         return nilai;
6     }
7
8     int absolute2(int nilai) {
9         if (nilai < 0)          // boleh tidak menggunakan kurung kurawal,
10            return -nilai;      // jika pernyataan dalam if hanya 1
11        return nilai;
12    }
```

2.4.2 if-then-else

Pengambilan keputusan yang berikutnya menggunakan struktur `if-then-else`. Jika ekspresi yang ada dalam `if` nilainya benar, maka blok pernyataan dalam `if` yang akan dieksekusi, sedangkan jika ekspresi nilainya salah, maka blok pernyataan dalam `else` yang akan dieksekusi.

Listing 7: Contoh if-then-else

```
    void checkLulus(float nilai) {
        if (nilai >= 60) {      // blok pernyataan dieksekusi jika nilai >= 60
            System.out.println("Lulus");
        }
        else {
            System.out.println("Tidak lulus");
        }
    }
```

Jika ekspresi yang ditest lebih dari satu (pengambilan keputusan berdasarkan beberapa kriteria), maka bisa digunakan struktur `if-else if- ... - else` seperti pada contoh berikut.

Listing 8: Contoh penggunaan if-then-else-if-...-else

```
/*
 * Contoh penggunaan if-else dengan banyak kondisi
 */
class ScorToGrade {
    public static void main(String[] args) {
        int testscore = 76;
        char grade;

        if (testscore >= 80) {
            grade = 'A';
        } else if (testscore >= 70) {
            grade = 'B';
        } else if (testscore >= 56) {
            grade = 'C';
        } else if (testscore >= 45) {
            grade = 'D';
        } else {
            grade = 'E';
        }
        System.out.println("Test score = " +
            testscore + ", Grade = " + grade);
    }
}
```

2.4.3 switch

Pengambilan keputusan berikutnya adalah dengan menggunakan `switch`. Perbedaan `if-then` dan `switch` adalah:

1. Pada `if-then` ekspresinya bisa 'lebih besar/lebih kecil/sama dengan', sedangkan `switch` expresinya 'sama dengan'.
2. Pada `if-then` hanya akan mengeksekusi 1 pernyataan blok, sedangkan `switch` memungkinkan untuk mengeksekusi beberapa pernyataan blok.

Kode berikut menunjukkan contoh penggunaan `switch`. Dalam contoh ini, variabel `month` ditest dalam `switch`, apabila sama dengan nilai tertentu pada rentang 1-12, maka akan dikonversikan ke nama bulannya. Sedangkan jika nilai `month` berada diluar rentang tersebut, akan diberikan nilai 'Bulan tidak valid'. Pada contoh berikut ini (Listing 9), keluaran dari program adalah `Agustus`.

Listing 9: Contoh penggunaan switch

```
/*
 * Konversi bulan dalam angka ke nama bulan.
 *   http://download.oracle.com/javase/tutorial/java/nutsandbolts/switch.html
 */
class SwitchDemo {
    public static void main(String[] args) {
        int month = 8;
        String monthString;
        switch (month) {
            case 1:  monthString = "Januari";       break;
            case 2:  monthString = "Februari";      break;
            case 3:  monthString = "Maret";         break;
            case 4:  monthString = "April";         break;
            case 5:  monthString = "Mei";           break;
            case 6:  monthString = "Juni";          break;
            case 7:  monthString = "Juli";          break;
            case 8:  monthString = "Agustus";       break;
            case 9:  monthString = "September";     break;
            case 10: monthString = "Oktober";       break;
            case 11: monthString = "November";      break;
            case 12: monthString = "Desember";      break;
            default: monthString = "Nilai bulan tidak valid"; break;
        }
        System.out.println(monthString);
    }
}
```

Pernyataan `break` setelah `case` dalam `switch` berfungsi untuk menghentikan eksekusi dari pernyataan-pernyataan selanjutnya. Jika `break` dihilangkan, maka apabila kondisi suatu `case` terpenuhi, pernyataan yang ada pada `case` berikutnya (walaupun kondisinya tidak terpenuhi), akan dieksekusi. Hal ini disebut dengan *fall through*. Hasil dari contoh program berikut (Listing 10) adalah `Agustus, September, Oktober, November, Desember` atau dengan kata lain, semua pernyataan setelah `case 8:` dieksekusi semuanya.

Listing 10: Contoh penggunaan switch (fall through)

```
/*
 * Konversi bulan dalam angka ke nama bulan.
 *   http://download.oracle.com/javase/tutorial/java/nutsandbolts/switch.html
 */
class SwitchDemoFallThrough {
    public static void main(String args[]) {
        java.util.ArrayList<String> futureMonths =
            new java.util.ArrayList<String>();
        int month = 8;

        switch (month) {
            case 1: futureMonths.add("Januari");
            case 2: futureMonths.add("Februari");
            case 3: futureMonths.add("Maret");
            case 4: futureMonths.add("April");
            case 5: futureMonths.add("Mei");
            case 6: futureMonths.add("Juni");
            case 7: futureMonths.add("Juli");
            case 8: futureMonths.add("Agustus");
```

```
            case 9: futureMonths.add("September");
            case 10: futureMonths.add("Oktober");
            case 11: futureMonths.add("November");
            case 12: futureMonths.add("Desember"); break;
            default: break;
        }

        if (futureMonths.isEmpty()) {
            System.out.println("Nilai bulan tidak valid");
        } else {
            for (String monthName : futureMonths) {
                System.out.println(monthName);
            }
        }
    }
}
```

`Switch` juga dapat digunakan dengan kondisi multipel `case`. Contoh penggunaan ini adalah pada program menghitung jumlah hari dalam sebulan. Jika bulan 1, 3, 5, 7, 8, 10, 12 maka jumlah harinya adalah 31, jika bulan 4, 6, 9, 11 maka jumlah harinya adalah 30, sedangkan untuk bulan 2, jika tahun kabisat jumlah harinya 29, selain itu jumlahnya 28 hari.

Listing 11: Contoh penggunaan switch (multiple case)

```
/*
 * Menghitung jumlah hari pada tahun dan bulan tertentu
 *   http://download.oracle.com/javase/tutorial/java/nutsandbolts/switch.html
 */
class NumDaysInMonth {
    public static void main(String[] args) {
        int month = 2;
        int year = 2000;
        int numDays = 0;

        switch (month) {
            case 1:
            case 3:
            case 5:
            case 7:
            case 8:
            case 10:
            case 12:
                numDays = 31;
                break;
            case 4:
            case 6:
            case 9:
            case 11:
                numDays = 30;
                break;
            case 2:
                if ( ((year % 4 == 0) && !(year % 100 == 0))
                     || (year % 400 == 0) )
                    numDays = 29;
                else
                    numDays = 28;
                break;
```

```
34              default:
35                  System.out.println("Nilai bulan tidak valid");
36                  break;
37          }
38          System.out.println("Jumlah hari = " + numDays);
39      }
40  }
```

2.4.4 `while`

Dalam pengulangan (*loop*), blok pernyataan akan dieksekusi selama hasil evaluasi dari ekspresi adalah *true*. Dalam bahasa Pemrograman Java, pengulangan dapat direalisasikan dengan: `while`, `do-while`, `for`. Pernyataan `while` mempunyai struktur seperti pada Listing 12, sedangkan contoh penggunaannya bisa dilihat pada Listing 13.

Listing 12: Konstruksi pengulangan dengan while

```
1       while(<ekspresi>) {
2           pernyataan-pernyataan
3       }
```

Listing 13: Contoh penggunaan while

```
1   class WhileDemo {
2       public static void main(String[] args){
3           int angka = 1, sum = 0;
4           while (angka <= 10) {
5               sum += angka;
6               angka++;
7           }
8           System.out.println("Sum = " + sum);
9       }
10  }
```

2.4.5 `do-while`

Pengulangan juga bisa diimplementasikan dengan `do-while`. Konstruksi `do-while` bisa dilihat di Listing 14. Perbedaan prinsip antara `while` dan `do-while` adalah urutan pengetesan kondisi (ekspresi) dan eksekusi pernyataan. Dalam `while`, ekspresi ditest dulu, baru blok pernyataan dieksekusi, sedangkan dalam `do-while` kebalikannya. Listing 14 adalah contoh penggunaan `do-while`.

Listing 14: Konstruksi pengulangan dengan do-while

```
1       do {
2           pernyataan-pernyataan
3       } while(<ekspresi>);
```

Listing 15: Contoh penggunaan do-while

```
1   class DoWhileDemo {
2       public static void main(String[] args){
3           int sum = 0, angka = 1;
4           do {
```

```
            sum += angka;
            angka ++;
        } while (angka <= 10);
        System.out.println("Sum = " + sum);
    }
}
```

2.4.6 `for`

Struktur pengulangan berikutnya adalah dengan menggunakan `for`. Konstruksinya bisa dilihat pada Listing 16 berikut.

Listing 16: Konstruksi pengulangan dengan for

```
    for (inisialisasi; terminasi; increment) {
        pernyataan.
    }
```

Pada `for`, proses eksekusi setiap ekspresi adalah sebagai berikut:

1. **inisialisasi** : expresi yang berisi kode inisialisasi. Hanya dieksekusi sekali saja pada saat mulai pengulangan.
2. **terminasi**: berisi kode pengetesan kondisi. Jika `true`, pengulangan akan diteruskan, jika `false`, pengulangan selesai.
3. **increment** : update kondisi, **dieksekusi setelah blok pernyataan dalam pengulangan**. Bisa *increment* atau *decrement*.

Contoh penggunaannya dapat dilihat pada Listing 17.

Listing 17: Contoh penggunaan for

```
class ForDemo {
    public static void main(String[] args){
        int sum = 0;
        for(int angka = 1; angka <= 10; angka++) {
            sum += angka;
        }
        System.out.println("Sum = " + sum);
    }
}
```

2.4.7 `continue`, `break` dan `return`

Dalam pengulangan, `continue` digunakan untuk **menskip** pernyataan, `break` digunakan untuk **menghentikan** pengulangan, sedangkan `return` digunakan untuk menghentikan pengulangan sekaligus keluar dari *method*. Berikut adalah contoh dari penggunaan `continue`, `break` dan `return` (Listing 18).

Listing 18: Contoh penggunaan continue, break, return

```
class ContinueBreakReturn {
    public static void main(String[] args){
        TestContinue();
        TestBreak();
        TestReturn();
```

```
6       }
7       /**
8        * Contoh penggunaan continue
9        */
10      public static void TestContinue() {
11          int angka = 6;
12          for (int n = 0; n < 10; n++) {
13              if (n == angka)
14                  continue;
15              System.out.print(n +  " ");
16          }
17          System.out.println("-- selesai --");
18      }
19      /**
20       * Contoh penggunaan break
21       */
22      public static void TestBreak() {
23          int angka = 6;
24          for (int n = 0; n < 10; n++) {
25              if (n == angka)
26                  break;
27              System.out.print(n +  " ");
28          }
29          System.out.println("-- selesai --");
30      }
31      /**
32       * Contoh penggunaan return
33       */
34      public static void TestReturn() {
35          int angka = 6;
36          for (int n = 0; n < 10; n++) {
37              if (n == angka)
38                  return;
39              System.out.print(n + " ");
40          }
41          System.out.println("-- selesai --");
42      }
43  }
```

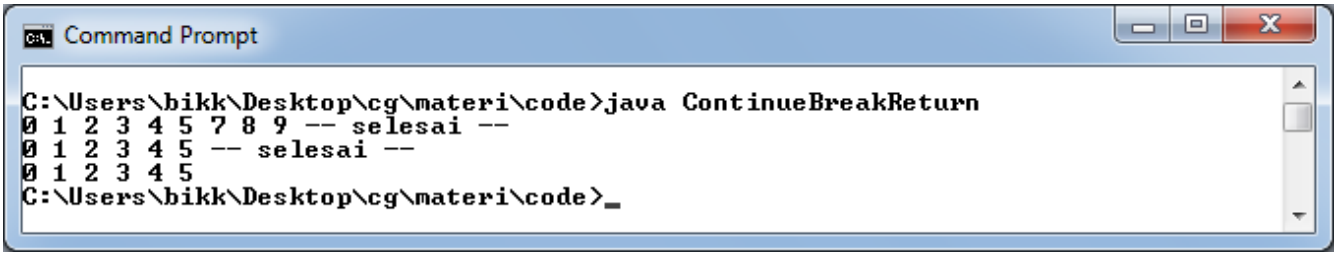

Figure 5: Hasil eksekusi program pada Listing 18

Hasil eksekusi dari program Listing 18 ditunjukkan pada Gambar 5. Perbedaan method `TestContinue`, `TestBreak`, `TestReturn` terletak hanya pada pernyataan `continue`, `break` dan `return`. Pernyataan `continue` mengabaikan angka 6 saja, `break` mengabaikan angka 6 s/d 9, `return` mengabaikan angka 6 s/d 9 dan `System.out.println("-- selesai --")`.

3 *Class* dan *Object*

3.1 Konsep Pemrograman Berorientasi Objek

Pemrograman berorientasi objek (OOP: *Object Oriented Programming*) adalah paradigma pemrograman yang berorientasikan objek. Dalam PBO, program dipandang sebagai kumpulan dari objek-objek yang saling berinteraksi satu sama lainnya.

Objek pada PBO, mengadopsi konsep objek pada dunia nyata. Yang dimaksud objek disini adalah segala sesuatu yang ada disekeliling kita yang mempunyai *state* dan *behavior*. Contoh objek yaitu TV, Mobil, Burung, Mahasiswa, Dosen, dan lain sebagainya. Untuk penjelasan, kita ambil contoh objek Burung. *State* pada burung misalnya: warna, berat, lapar. Sedangkan *behavior* menggambarkan perilaku dari burung misalnya: terbang, berjalan dan sebagainya. *State* biasanya digambarkan dengan kata benda dan kata sifat, sedangkan *behaviour* digambarkan dengan kata kerja.

Secara konseptual, objek pada program juga sama dengan objek pada dunia nyata yaitu memiliki *state* dan *behavior*. *State* dari sebuah objek disimpan di dalam *fields* (atau bisa juga disebut variabel), sedangkan *behavior* diekspose melalui *method* (atau ada yang menyebutnya fungsi). Keuntungan dari PBO adalah:

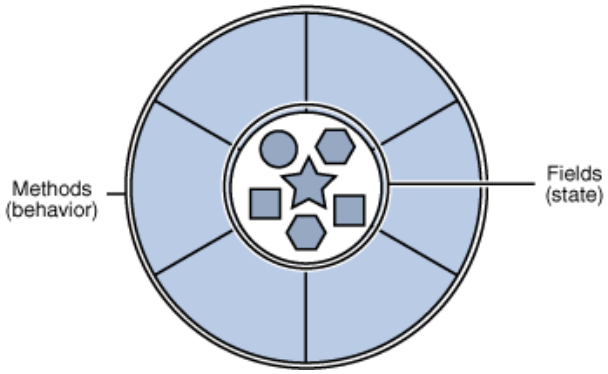

Figure 6: Konsep Objek

1. Modularity
2. Information-hidding
3. Code re-use
4. Pluggability dan kemudahan debugging.

3.1.1 Class

Didunia nyata, sering kita jumpai banyak objek dari jenis yang sama, contohnya ada banyak mahasiswa, ada banyak TV, ada banyak mobil. Setiap objek yang tipenya sama memiliki komponen ataupun *blueprint* yang sama pula. Dalam terminologi PBO, contohnya mobil pribadi anda adalah *instance* dari *class* Mobil. Atau dengan kata lain, sebuah *class* adalah *blueprint* untuk mencetak objek-objek. Berikut contoh class.

Listing 19: Contoh class Mobil

```
1 public class Mobil {
2     int kecepatan = 0;
3     String jenis = "";
```

```
4
5       public void SetJenis(String str) {
6           jenis = str;
7       }
8       public void Akselerasi(int akselerasi) {
9           kecepatan += akselerasi;
10      }
11      public void Rem(int deakselerasi) {
12          kecepatan -= deakselerasi;
13      }
14      public void Tampilkan() {
15          System.out.println("Mobil: " + jenis + ", kecepatan: " + kecepatan);
16      }
17  }
```

Berikut adalah contoh class `MobilTest` yang membuat dua objek mobil (`mobilSaya` dan `mobilAnda`).

Listing 20: Contoh class MobilTest

```
1  class MobilTest {
2      public static void main(String[] args) {
3          //buat instance mobil
4          Mobil mobilSaya = new Mobil();
5          Mobil mobilAnda = new Mobil();
6
7          //panggil beberapa method
8          mobilSaya.SetJenis("SUV");
9          mobilSaya.Akselerasi(50);
10         mobilSaya.Tampilkan();
11
12         mobilAnda.SetJenis("Sedan");
13         mobilAnda.Akselerasi(60);
14         mobilAnda.Tampilkan();
15     }
16 }
```

Hasil eksekusi dari program pada Listing 20 adalah:

```
Mobil: SUV, kecepatan: 50
Mobil: Sedan, kecepatan: 60
```