Modul 4 Looping

Tugas Pendahuluan

1. Jelaskan perbedaan antara loop for, while, dan do-while dalam Java. Berikan contoh penggunaan masing-masing loop.
2. Apa yang dimaksud dengan loop bersarang (nested loop)? Berikan contoh program Java yang menggunakan loop bersarang.
3. Jelaskan perbedaan antara operator ++i dan i++ dalam Java. Berikan contoh penggunaan keduanya.

Materi

1. Looping Control

Dalam bahasa pemrograman Java, terdapat beberapa struktur pengulangan (loop control) yang digunakan untuk mengulang blok kode tertentu secara berulang hingga kondisi tertentu terpenuhi. Berikut adalah beberapa struktur pengulangan yang umum digunakan dalam Java:

1) Loop for:
   Digunakan untuk mengulang blok kode dengan jumlah pengulangan yang telah ditentukan sebelumnya. Terdiri dari inisialisasi variabel pengontrol, kondisi penghentian, dan perubahan nilai variabel pengontrol.
   Contoh:

```
for (int i = 0; i < 5; i++) {
    // Blok kode yang akan diulang
    System.out.println(i);
}
```

2) Loop while:
   Digunakan untuk mengulang blok kode selama kondisi tertentu terpenuhi. Terdiri dari kondisi penghentian sebelum blok kode dieksekusi.

   Contoh:

```
int i = 0;
while (i < 5) {
    // Blok kode yang akan diulang
    System.out.println(i);
    i++;
}
```

3) Loop do-while:
   Mirip dengan loop while, tetapi blok kode dieksekusi setidaknya satu kali sebelum pengecekan kondisi. Kondisi penghentian ditempatkan setelah blok kode.

Contoh:

```
int i = 0;
do {
    // Blok kode yang akan diulang
    System.out.println(i);
    i++;
} while (i < 5);
```

4) Loop foreach (untuk koleksi):
Digunakan untuk mengulang elemen-elemen dalam suatu koleksi (array, list, dll.) tanpa menggunakan indeks. Cocok untuk mengakses setiap elemen dalam koleksi secara berurutan.

Contoh:

```
int[] numbers = {1, 2, 3, 4, 5};
for (int number : numbers) {
    // Blok kode yang akan diulang
    System.out.println(number);
}
```

Struktur pengulangan ini memungkinkan Anda untuk mengatur pengulangan berdasarkan kondisi tertentu, sehingga memungkinkan pengulangan yang fleksibel dan dinamis dalam program Java Anda. Selain itu, terdapat juga pernyataan break dan continue yang dapat digunakan dalam pengulangan untuk menghentikan pengulangan atau melanjutkan ke iterasi berikutnya berdasarkan kondisi tertentu.

Catatan: Pastikan untuk menggunakan struktur pengulangan yang sesuai dengan kebutuhan dan menjaga agar kondisi penghentian tidak menyebabkan pengulangan tak terbatas (infinite loop).

2. Loop Control Statements

Dalam bahasa pemrograman Java, terdapat beberapa pernyataan pengendalian loop (loop control statements) yang digunakan untuk mengontrol aliran eksekusi dalam struktur pengulangan. Berikut adalah beberapa pernyataan pengendalian loop yang umum digunakan:

1) Pernyataan break:
Digunakan untuk menghentikan eksekusi loop secara paksa ketika kondisi tertentu terpenuhi. Saat pernyataan break dieksekusi, program akan keluar dari loop dan melanjutkan eksekusi pada pernyataan setelah loop.

Contoh:

```
for (int i = 0; i < 10; i++) {
    if (i == 5) {
        break; // Keluar dari loop ketika i = 5
    }
    System.out.println(i);
}
```

2) Pernyataan continue:

Digunakan untuk melompati sisa blok kode dalam satu iterasi loop saat kondisi tertentu terpenuhi. Saat pernyataan continue dieksekusi, program akan melanjutkan ke iterasi berikutnya tanpa menjalankan kode di bawahnya dalam iterasi saat ini.

Contoh:

```
for (int i = 0; i < 10; i++) {
    if (i % 2 == 0) {
        continue; // Lewatkan iterasi saat i adalah bilangan genap
    }
    System.out.println(i);
}
```

3) Pernyataan return (dalam metode):

Digunakan untuk menghentikan eksekusi metode dan mengembalikan nilai ke pemanggil metode. Saat pernyataan return dieksekusi, program keluar dari metode dan melanjutkan eksekusi pada pernyataan setelah pemanggil metode.

Contoh:

```
public int hitungJumlah(int a, int b) {
    int jumlah = a + b;
    return jumlah; // Mengembalikan nilai jumlah dan menghentikan eksekusi metode
}
```

Pernyataan pengendalian loop ini memberikan fleksibilitas dalam mengontrol aliran eksekusi dalam pengulangan. Dengan menggunakan pernyataan break, continue, atau return, Anda dapat memutuskan untuk keluar dari loop, melanjutkan ke iterasi berikutnya, atau bahkan menghentikan eksekusi metode.

Catatan: Pastikan untuk menggunakan pernyataan pengendalian loop dengan hati-hati dan hanya saat diperlukan agar tidak mempengaruhi logika program secara keseluruhan.

3. Enhanced for loop in Java.

Pada bahasa pemrograman Java, terdapat struktur pengulangan yang disebut enhanced for loop atau juga dikenal dengan "for-each" loop. Enhanced for loop menyediakan cara yang lebih sederhana untuk melakukan iterasi pada elemen-elemen dalam array atau koleksi tanpa menggunakan variabel indeks. Enhanced for loop ini digunakan secara khusus untuk mengiterasi elemen-elemen secara berurutan.

Berikut adalah sintaks dari enhanced for loop:

```
for (tipeDataElemen elemen : koleksi) {
    // Kode yang akan dieksekusi untuk setiap elemen
}
```

Berikut penjelasan mengenai komponen-komponen dari enhanced for loop:

1) tipeDataElemen: Tipe data dari elemen-elemen dalam koleksi atau array.
2) elemen: Variabel yang merepresentasikan setiap elemen dalam koleksi atau array pada setiap iterasi.
3) koleksi: Array atau koleksi yang ingin diiterasi.

Enhanced for loop secara otomatis mengatur iterasi dan pengambilan elemen-elemen dari koleksi atau array. Hal ini menghilangkan kebutuhan untuk secara manual mengelola indeks atau menggunakan variabel penghitung terpisah. Loop ini akan mengiterasi melalui setiap elemen koleksi, menugaskan elemen tersebut ke variabel elemen, dan menjalankan blok kode dalam loop untuk setiap iterasi.

Berikut contoh penggunaan enhanced for loop dengan array:

```
int[] angka = {1, 2, 3, 4, 5};

for (int nilai : angka) {
    System.out.println(nilai);
}
```

Pada contoh tersebut, enhanced for loop akan mengiterasi melalui setiap elemen dalam array angka. Pada setiap iterasi, elemen saat ini akan ditugaskan ke variabel nilai, dan nilainya akan dicetak. Hal ini menyederhanakan proses iterasi pada array dan mengakses setiap elemen.

Enhanced for loop juga dapat digunakan dengan tipe koleksi lainnya, seperti list, set, dan map. Selama koleksi tersebut mengimplementasikan interface Iterable, maka dapat digunakan dalam enhanced for loop.

Enhanced for loop merupakan cara yang praktis dan ringkas untuk mengiterasi elemen-elemen dalam array atau koleksi. Hal ini meningkatkan keterbacaan kode dan mengurangi peluang terjadinya kesalahan indeks yang umum terjadi pada penggunaan for loop tradisional.

4. Perbedaan antara do while dan while berikut contohnya :

Perbedaan antara do-while dan while terletak pada urutan evaluasi kondisi. Berikut adalah perbedaan antara keduanya:

do-while:

1) Evaluasi kondisi dilakukan setelah menjalankan blok kode.
2) Blok kode akan dieksekusi setidaknya satu kali, bahkan jika kondisi awal tidak terpenuhi.
3) Cocok digunakan jika Anda ingin menjalankan blok kode terlebih dahulu sebelum memeriksa kondisi.
4) Kondisi penghentian terletak pada akhir loop.

while:

1) Evaluasi kondisi dilakukan sebelum menjalankan blok kode.
2) Blok kode mungkin tidak dieksekusi sama sekali jika kondisi awal tidak terpenuhi.
3) Cocok digunakan jika Anda ingin memeriksa kondisi sebelum menjalankan blok kode.
4) Kondisi penghentian terletak pada awal loop.

Berikut adalah contoh program untuk membandingkan penggunaan do-while dan while dalam menghitung jumlah bilangan dari 1 hingga 5:

Contoh dengan do-while:

```
1. public class ContohDoWhile {
2.     public static void main(String[] args) {
3.         int i = 1;
4.         int jumlah = 0;
5.
6.         do {
7.             jumlah += i;
8.             i++;
9.         } while (i <= 5);
10.
11.        System.out.println("Jumlah bilangan: " + jumlah);
12.    }
13. }
14.
```

Contoh dengan while:

```
1. public class ContohWhile {
2.     public static void main(String[] args) {
3.         int i = 1;
4.         int jumlah = 0;
5.
6.         while (i <= 5) {
7.             jumlah += i;
8.             i++;
9.         }
10.
11.        System.out.println("Jumlah bilangan: " + jumlah);
12.    }
13. }
14.
```

Kedua contoh program di atas menghitung jumlah bilangan dari 1 hingga 5. Perhatikan bahwa di contoh do-while, blok kode yang melakukan penjumlahan dieksekusi terlebih dahulu sebelum kondisi i <= 5 dievaluasi. Sedangkan di contoh while, kondisi i <= 5 dievaluasi terlebih dahulu sebelum blok kode dieksekusi.

Output dari kedua program ini adalah:

```
Jumlah bilangan: 15
```

Meskipun outputnya sama, namun urutan evaluasi dan perilaku eksekusi loop berbeda antara do-while dan while. Pilihlah yang paling sesuai dengan kebutuhan dan logika program Anda.

5. Kapan harus kita menggunakan for

Loop for biasanya digunakan ketika Anda mengetahui berapa kali pengulangan harus dilakukan sebelumnya atau memiliki jumlah iterasi yang tetap. Beberapa situasi di mana Anda dapat menggunakan loop for antara lain:

1) Mengulang sejumlah kali yang sudah diketahui:
   Ketika Anda perlu melakukan pengulangan sejumlah kali yang sudah diketahui sebelumnya, seperti mencetak elemen-elemen dalam array atau melakukan iterasi sebanyak N kali.

2) Iterasi melalui array atau koleksi:
   Ketika Anda ingin mengiterasi melalui elemen-elemen dalam array, list, set, atau koleksi lainnya, menggunakan indeks atau iterator.

3) Menggunakan variabel pengontrol pengulangan:
   Ketika Anda ingin menggunakan variabel pengontrol pengulangan untuk mengontrol proses iterasi, seperti mengubah nilai variabel pengontrol atau melompati beberapa iterasi dengan menggunakan pernyataan continue atau break.

4) Melakukan perulangan berdasarkan kondisi kompleks:
   Ketika Anda perlu mengatur kondisi penghentian pengulangan yang kompleks, dengan menggunakan pernyataan logika yang lebih rumit di dalam bagian pengontrol pengulangan.

5) Pengulangan berurutan:
   Ketika Anda ingin melakukan pengulangan berurutan dengan mengikuti langkah yang terdefinisi, seperti penambahan atau pengurangan nilai variabel pengontrol setiap iterasi.

Dalam banyak kasus, loop for memberikan sintaks yang lebih sederhana dan terstruktur, dan mudah untuk mengontrol iterasi. Namun, jika jumlah iterasi tidak diketahui sebelumnya atau Anda perlu mengulang dengan kondisi yang lebih fleksibel, maka loop while atau do-while mungkin lebih cocok.

Pilihan antara loop for, while, atau do-while tergantung pada kebutuhan dan logika program yang sedang Anda kembangkan.

6. Nested Loop

Loop bersarang adalah penggunaan satu atau lebih loop di dalam loop lainnya. Dalam konteks Java, ini berarti memiliki satu loop di dalam blok kode lainnya. Loop bersarang memungkinkan pengulangan yang lebih kompleks dan fleksibel dengan menggunakan iterasi bersarang.

Berikut adalah contoh program Java yang menggunakan loop bersarang untuk mencetak pola bintang segitiga terbalik:

```
 1. public class PolaBintangBersarang {
 2.     public static void main(String[] args) {
 3.         int tinggi = 5;
 4.
 5.         for (int i = tinggi; i >= 1; i--) {
 6.             for (int j = 1; j <= i; j++) {
 7.                 System.out.print("* ");
 8.             }
 9.             System.out.println();
10.         }
11.     }
12. }
13.
```

Pada contoh di atas, loop for luar digunakan untuk mengatur jumlah baris pada pola bintang segitiga terbalik. Loop for dalam digunakan untuk mencetak bintang pada setiap baris.

Output dari program ini adalah pola bintang segitiga terbalik seperti berikut:

```
* * * * *
* * * *
* * *
* *
*
```

Dalam program ini, variabel tinggi menentukan jumlah baris yang ingin dicetak pada pola. Loop for luar akan terus berjalan hingga mencapai baris terakhir, dan loop for dalam akan mencetak bintang pada setiap baris tergantung pada nilai variabel i.

Loop bersarang dapat digunakan dalam berbagai situasi, seperti ketika perlu melakukan pengulangan dalam pengulangan, melakukan pencarian atau pemrosesan pada elemen-elemen dalam matriks dua dimensi, atau mencari kombinasi tertentu dalam data yang kompleks. Namun, penting untuk diperhatikan bahwa penggunaan loop bersarang harus dilakukan dengan hati-hati dan dipertimbangkan dengan baik, karena dapat menyebabkan kompleksitas dan kinerja program yang buruk jika tidak dikelola dengan benar.

Berikut adalah contoh program Java yang menggunakan loop bersarang dengan kombinasi for dan while untuk mencetak pola angka segitiga:

```
 1. public class PolaAngkaBersarang {
 2.     public static void main(String[] args) {
 3.         int tinggi = 5;
 4.         int baris = 1;
 5.
 6.         for (; baris <= tinggi; baris++) {
 7.             int angka = 1;
 8.
 9.             while (angka <= baris) {
10.                 System.out.print(angka + " ");
11.                 angka++;
12.             }
13.
14.             System.out.println();
15.         }
16.     }
17. }
18.
```

Pada contoh di atas, loop for digunakan untuk mengatur jumlah baris pada pola angka segitiga. Loop while digunakan di dalam loop for untuk mencetak angka pada setiap baris.

Output dari program ini adalah pola angka segitiga seperti berikut:

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Dalam program ini, variabel tinggi menentukan jumlah baris yang ingin dicetak pada pola. Loop for luar akan terus berjalan hingga mencapai baris terakhir. Di dalam loop for, variabel angka diinisialisasi dengan nilai 1 dan akan terus dicetak hingga mencapai nilai baris.

Loop bersarang dengan kombinasi for dan while dapat digunakan untuk mengatasi situasi di mana Anda membutuhkan pengulangan kompleks dengan logika yang berbeda-beda. Pastikan untuk mengatur kondisi penghentian loop dengan benar agar program tidak masuk ke dalam pengulangan tak terbatas.

Contoh Program

Contoh 1:

Berikut adalah contoh program Java yang menggunakan kombinasi berbagai jenis loop untuk mencetak pola bintang segitiga:

```
 1. public class PolaBintang {
 2.     public static void main(String[] args) {
 3.         int tinggi = 5;
 4.
 5.         // Loop for luar untuk mengatur jumlah baris
 6.         for (int i = 1; i <= tinggi; i++) {
 7.
 8.             // Loop for dalam untuk mencetak bintang pada setiap baris
 9.             for (int j = 1; j <= i; j++) {
10.                 System.out.print("* ");
11.             }
12.
13.             // Mencetak baris baru setelah mencetak bintang pada setiap baris
14.             System.out.println();
15.         }
16.     }
17. }
18.
```

Pada contoh di atas, digunakan kombinasi loop for luar dan for dalam untuk mencetak pola bintang segitiga. Loop for luar digunakan untuk mengatur jumlah baris yang ingin dicetak, sedangkan loop for dalam digunakan untuk mencetak bintang pada setiap baris.

Output dari program ini adalah pola bintang segitiga seperti berikut:

```
*
* *
* * *
* * * *
* * * * *
```

Dalam program ini, variabel tinggi digunakan untuk menentukan tinggi segitiga, yaitu jumlah baris yang ingin dicetak. Program akan mencetak bintang pada setiap baris sebanyak nomor baris tersebut.

Dengan menggunakan kombinasi loop seperti ini, Anda dapat membuat pola-pola lain atau menggabungkan dengan logika-program lainnya sesuai dengan kebutuhan Anda.

Contoh 2 :

Berikut adalah contoh program Java yang menggunakan loop while untuk mencetak bilangan dari 1 hingga 10:

```
1. public class CetakBilangan {
2.     public static void main(String[] args) {
3.         int i = 1;
4.
5.         while (i <= 10) {
6.             System.out.println(i);
7.             i++;
8.         }
9.     }
10. }
11.
```

Pada contoh di atas, digunakan loop while untuk mencetak bilangan dari 1 hingga 10. Loop akan terus berjalan selama kondisi i <= 10 terpenuhi. Pada setiap iterasi, program akan mencetak nilai i dan kemudian menambahkan nilai i dengan 1 menggunakan operator ++.

Output dari program ini adalah sebagai berikut:

```
1
2
3
4
5
6
7
8
9
10
```

Dalam program ini, variabel i diinisialisasi dengan nilai awal 1 sebelum memasuki loop. Kemudian, pada setiap iterasi, nilai i dicetak dan diinkremenkan (ditambah 1). Loop akan terus berjalan hingga nilai i mencapai atau melebihi 10.

Anda dapat mengganti kondisi loop while sesuai kebutuhan, serta menambahkan logika-program lainnya di dalam loop tersebut.

Contoh 3

Berikut adalah contoh program Java yang menggunakan loop do-while untuk meminta pengguna memasukkan bilangan positif:

```
1. import java.util.Scanner;
```

```
 2.
 3. public class InputBilanganPositif {
 4.     public static void main(String[] args) {
 5.         Scanner input = new Scanner(System.in);
 6.         int bilangan;
 7.
 8.         do {
 9.             System.out.print("Masukkan bilangan positif: ");
10.             bilangan = input.nextInt();
11.         } while (bilangan <= 0);
12.
13.         System.out.println("Bilangan positif yang dimasukkan: " + bilangan);
14.     }
15. }
16.
```

Pada contoh di atas, digunakan loop do-while untuk meminta pengguna memasukkan bilangan positif. Program akan terus meminta input bilangan dari pengguna menggunakan Scanner, dan akan terus mengulang pengulangan jika bilangan yang dimasukkan kurang dari atau sama dengan 0.

Output dari program ini adalah bilangan positif yang dimasukkan oleh pengguna.

Contoh jalannya program:

```
Masukkan bilangan positif: -5
Masukkan bilangan positif: 0
Masukkan bilangan positif: 10
Bilangan positif yang dimasukkan: 10
```

Dalam program ini, loop do-while memastikan bahwa setidaknya satu kali iterasi akan dilakukan sebelum kondisi penghentian dievaluasi. Artinya, program akan meminta pengguna memasukkan bilangan minimal satu kali, dan setelah itu akan memeriksa apakah bilangan yang dimasukkan positif. Jika tidak, program akan terus meminta input hingga bilangan positif yang valid dimasukkan oleh pengguna.

Anda dapat mengubah kondisi penghentian loop do-while sesuai kebutuhan, serta menambahkan logika-program lainnya di dalam loop tersebut.

Latihan

Berikut adalah beberapa contoh program dengan kesalahan dalam penggunaan loop. Tugas Anda adalah memperbaiki program-program tersebut sehingga dapat berjalan dengan semestinya. Anda perlu mengidentifikasi kesalahan yang ada dan melakukan perbaikan sesuai dengan logika yang benar.

1. Program: Cetak Bilangan Genap

```
1. public class CetakBilanganGenap {
 2.     public static void main(String[] args) {
 3.         int n = 10;
 4.         for (int i = 1; i <= n; i++) {
 5.             if (i % 2 == 0) {
 6.                 System.out.println(i);
 7.             }
 8.         }
 9.     }
10. }
```

```
11.
```

Program di atas bertujuan untuk mencetak bilangan genap dari 1 hingga 10, tetapi tidak mencetak apa pun. Perbaiki program tersebut agar dapat mencetak bilangan genap dengan benar.

2. Program: Jumlah Deret

```
1. public class JumlahDeret {
 2.     public static void main(String[] args) {
 3.         int n = 5;
 4.         int jumlah = 0;
 5.         while (int i = 1; i <= n) {
 6.             jumlah += i;
 7.             i++;
 8.         }
 9.         System.out.println("Jumlah deret: " + jumlah);
10.     }
11. }
12.
```

Program di atas bertujuan untuk menghitung jumlah deret bilangan dari 1 hingga 5, tetapi terdapat kesalahan dalam penggunaan loop while. Perbaiki program tersebut agar dapat menghitung jumlah deret dengan benar.

3. Program: Cetak Pola Bintang

```
 1. public class CetakPolaBintang {
 2.     public static void main(String[] args) {
 3.         int n = 5;
 4.         for (int i = n; i >= 1; i++) {
 5.             for (int j = 1; j <= i; j++) {
 6.                 System.out.print("* ");
 7.             }
 8.             System.out.println();
 9.         }
10.     }
11. }
12.
```

Program di atas bertujuan untuk mencetak pola bintang segitiga terbalik, tetapi menghasilkan kesalahan pada loop for luar. Perbaiki program tersebut agar dapat mencetak pola bintang dengan benar.

4. Program: Perkalian Tabel

```
 1. public class TabelPerkalian {
 2.     public static void main(String[] args) {
 3.         for (int i = 1; i <= 10; i++) {
 4.             for (int j = 1; i <= 10; j++) {
 5.                 System.out.print(i * j + " ");
 6.             }
 7.             System.out.println();
 8.         }
 9.     }
10. }
11.
```

Program di atas bertujuan untuk mencetak tabel perkalian dari 1 hingga 10, tetapi terdapat kesalahan pada loop for dalam. Perbaiki program tersebut agar dapat mencetak tabel perkalian dengan benar.

Instruksi:

4) Perbaiki kesalahan yang ada dalam setiap program.
5) Jalankan program-program yang telah diperbaiki untuk memastikan bahwa program berjalan dengan semestinya.
6) Tulislah kode program yang telah diperbaiki dalam file dengan ekstensi .java.

Tugas

1. Buatlah sebuah program Java yang menggunakan loop for untuk mencetak deret bilangan genap dari 2 hingga 20.
2. Buatlah sebuah program Java yang menggunakan loop while untuk mencetak deret bilangan ganjil dari 1 hingga 15.
3. Buatlah sebuah program Java yang menggunakan loop do-while untuk meminta pengguna memasukkan bilangan positif, dan terus meminta hingga pengguna memasukkan bilangan negatif. Setelah itu, program mencetak jumlah bilangan positif yang dimasukkan.
4. Buatlah sebuah program Java yang menggunakan loop bersarang untuk mencetak pola bintang segitiga terbalik:

```
*****
****
***
**
*
```

Buatlah sebuah program Java yang menggunakan loop bersarang untuk mencetak pola angka segitiga:

```
1
12
123
1234
12345
```

Instruksi:

1) Tulislah jawaban untuk tugas teori dalam bentuk tulisan atau dokumen.
2) Implementasikan program-program praktikum dalam file-file dengan ekstensi .java.
3) Jalankan program-program untuk memastikan tidak ada kesalahan.