Tugas 1

Nama & NPM	Topik:	Tanggal:
Abdullah Hasyim Syauqi G1F024019	**Unit 1 FOR**	**16-09-2024**

Identifikasi Masalah:

1.1 Evaluasi penyebab kesalahan dan perbaiki kode pada Contoh 1!

Jawaban : 1. Posisi inisialisasi dan kondisi terbalik di dalam pernyataan for.
2. Kesalahan tipe data double untuk variabel y.

1.2 Rekomendasikan kode yang tepat menggunakan break atau continue terhadap pertama atau kedua agar menghasilkan luaran berikut:
Luaran Contoh 2:

```
i = 1; j = 1
i = 1; j = 2
i = 2; j = 1
i = 2; j = 2
```

Jawaban : Berdasarkan luaran yang diinginkan kita perlu menggunakan kata kunci **break** atau **continue** pada baris kode kosong untuk mengontrol alur perulangan.

1.3 Cermati kode contoh 3. Apabila ingin menghasilkan luaran berikut:
Luaran berbentuk piramida
Masukan Input: 7

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

Rekomendasikan kode untuk menghasilkan luaran tersebut!

Jawaban : 1. Perulangan untuk spasi: Setiap baris harus dimulai dengan sejumlah spasi yang menurun dari baris pertama ke baris terakhir.
2. Perulangan untuk bintang: Jumlah bintang bertambah 2 di setiap baris, dimulai dari 1.

1.4 Analisa diagram flowchart dari Latihan 1.2 dan 1.3!

Penyusunan Algoritma dan Kode Program

1.1

```
public class ContohFor{
   public static void main(String[] args) {
      // Inisialisasi dengan tipe data int
      for (int y = 0; y <= 15; y++) {
         if (y % 2 == 0) {
            System.out.println(y + " ");
         } else if (y == 8) {
         }
      }
   }
}
```

Rekomendasi kata kunci pada baris kosong :
Baris Kode 1: Letakkan System.out.println(y + " "); untuk mencetak nilai y yaitu bilangan genap.
Baris Kode 2: Tidak perlu perubahan karena keluaran yang diinginkan tidak melibatkan kondisi y == 8.

1.2

```
public class ForBersarang {
   public static void main(String[] args) {
      pertama:
      for (int i = 1; i < 5; i++) {
         kedua:
         for (int j = 1; j < 3; j++) {
            System.out.println("i = " + i + "; j = " + j);
         }
         if (i == 2) {
            break pertama;
         }
      }
   }
}
```

- Pertama : Menandai perulangan for pertama, yang mengontrol variabel i.
- break pertama ; Saat **i == 2**, kita menghentikan perulangan yang diberi label **pertama**. Ini berarti perulangan tidak akan dilanjutkan ke nilai **i = 3** dan seterusnya.

1.3

```
import java.util.Scanner;

public class ForBersarang {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      System.out.print("Masukan Input: ");
      int tinggi = input.nextInt();

      for (int t = 1; t <= tinggi; t++) {
         for (int s = tinggi; s > t; s--) {
            System.out.print(" ");
         }

         for (int b = 1; b <= (2 * t - 1); b++) {
            System.out.print("*");
         }

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

1.4

A. Analisa flowchart latihan 1.2

1. Mulai: awal program.
2. Inisialisasi variabel: Di sini, variabel loop diinisialisasi (misalnya, int i = 0).
3. Cek kondisi: Kondisi di dalam perulangan for diperiksa (misalnya, apakah i masih lebih kecil dari nilai maksimum).

4. Eksekusi blok kode: Jika kondisi benar, blok kode di dalam perulangan akan dijalankan.
5. Increment atau decrement: Setelah eksekusi, variabel loop ditambah atau dikurangi.
6. Perulangan kembali ke cek kondisi: Jika kondisi masih memenuhi, proses kembali ke langkah 3.
7. Akhir loop: Jika kondisi tidak terpenuhi, perulangan berhenti.
8. Selesai: Program berakhir.

Dalam flowchart ini, kita melihat alur yang sangat jelas antara pemeriksaan kondisi, eksekusi, dan pengulangan. Alur perulangan berulang sampai kondisi yang ditentukan tidak lagi terpenuhi

B. Analisa flowchart latihan 1.3

1. Mulai: Representasi awal program.
2. Inisialisasi variabel luar (loop luar): Misalnya, int i = 0 untuk loop pertama.
3. Cek kondisi loop luar: Memeriksa apakah kondisi perulangan luar masih terpenuhi.
4. Inisialisasi variabel dalam (loop dalam): Setelah loop luar masuk ke iterasinya, loop dalam diinisialisasi (misalnya, int j = 0).
5. Cek kondisi loop dalam: Loop dalam memeriksa kondisinya dan menjalankan blok kode di dalamnya jika terpenuhi.
6. Eksekusi blok kode dalam loop dalam: Eksekusi proses di dalam loop dalam (misalnya, mencetak bintang dalam piramida).
7. Increment loop dalam: Loop dalam ditambah (misalnya, j++).
8. Perulangan kembali ke cek kondisi loop dalam: Jika kondisi loop dalam masih terpenuhi, proses kembali ke langkah 5.
9. Increment loop luar: Setelah loop dalam selesai, loop luar ditambah (misalnya, i++).
10. Perulangan kembali ke cek kondisi loop luar: Jika kondisi loop luar masih terpenuhi, proses kembali ke langkah 3.
11. Selesai: Program berakhir setelah semua perulangan selesai.

Pola piramida dibuat dengan menggunakan pengulangan bersarang, di mana perulangan dalam mengendalikan jumlah elemen (misalnya, jumlah bintang atau spasi) yang dicetak dalam setiap iterasi perulangan luar.

Kesimpulan

saya melakukan perbaikan pada kode Java yang menggunakan perulangan for dan perulangan bersarang, dengan fokus pada penggunaan yang tepat dari struktur kontrol seperti break untuk mengatur alur program. saya juga berhasil menciptakan pola piramida dengan pengulangan, serta menganalisis flowchart untuk memahami alur logika perulangan dengan lebih baik.

Refleksi

pentingnya pemahaman mendalam tentang struktur perulangan dalam pemrograman, yang memungkinkan kita untuk menulis kode yang lebih efisien dan terorganisir.

Tugas 2

Nama & NPM	Topik:	Tanggal:
Abdullah Hasyim Syauqi G1F024019	**Unit 2 WHILE**	**16-09-2024**

Identifikasi Masalah:

2.1 Ubahlah baris kode pada Contoh 4

//Ubah1 menjadi if(i % 3 == 0){ ◊ running, periksa hasilnya

//Ubah2 menjadi continue; ◊ running, periksa hasilnya

Evaluasi perbandingan luaran sebelum dan setelah diubah! Simpulkan maksud dari perubahan tersebut!

Jawaban : ada di penyusunan algoritma dan kode program dibawah

2.2 cermati Contoh 5. Periksa luaran, bila ketika di eksekusi, jumlah yang diulang = 0!

Evaluasi luaran, bila kode diubah menjadi do … while dengan masukan sama jumlah yang diulang = 0.

Simpulkan perbedaan while dan do … while!

Jawaban : ada di penyusunan algoritma dan kode program dibawah

2.3 Bila diketahui pernyataan pseudocode berikut:

[1] inisiasi idPelajaran
[2] inisiasi nilai pelajaran
[3] inisiasi nilai rata-rata
[4] Minta pengguna untuk menuliskan jumlah pelajaran
[5] Ketika idPelajaran lebih kecil dari jumlah pelajaran
[6] Minta pengguna untuk menuliskan nilai pelajaran
[7] Hitung nilai rata-rata = (nilai pelajaran + nilai rata-rata) / 2
[8] Tambah satu ke idPelajaran
[9] Tampilkan nilai rata-rata

Rekomendasikan kode untuk menyelesaikan Pseudocode tersebut!

2.4 Rancang diagram flowchart dari Latihan 2.1, Latihan 2.2, dan Latihan 2.3!

Penyusunan Algoritma dan Kode Program

2.1

Kode setelah ubah 1:

```
public class ContohWhile {
   public static void main(String[] args) {
      int i = 1;
      while (i <= 6) {
         System.out.println(i);
         i++;
         if (i % 3 == 0) {
            break;
         }
      }
   }
}
```

Kode setelah ubah 2 :

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

```
        int i = 1;
        while (i <= 6) {
            System.out.println(i);
            i++;
            if (i % 3 == 0) {
                continue;
            }
        }
    }
}
```

Setelah Ubah1:

Dengan menggunakan if(i % 3 == 0), perulangan tetap berlanjut hingga mencapai angka 4. Program mencetak 1, 2, dan 3, lalu menghentikan proses karena i menjadi 4, tetapi efeknya sama dengan sebelum diubah karena break tetap menghentikan perulangan.

Setelah Ubah2:

Mengganti break; dengan continue; membuat program melewatkan iterasi ketika i mencapai angka yang merupakan kelipatan 3. Jadi, pada saat i bernilai 3, program mencetak 1 dan 2, lalu ketika i bernilai 3, ia melanjutkan ke iterasi berikutnya, yang membuat 4, 5, dan 6 tetap dicetak.

2.2

```
import java.util.Scanner;

public class ForBersarang {
    public static void main(String[] args) {
        Scanner dataKata = new Scanner(System.in);
        System.out.print("Masukkan Kata yang ingin diulang : ");
        String kata = dataKata.nextLine();

        Scanner dataJumlah = new Scanner(System.in);
        System.out.print("Masukkan Jumlah ingin diulang : ");
        int jumlah = dataJumlah.nextInt();

        int i = 0;
        do {
            System.out.println(kata);
            i++;
        } while(i < jumlah);
    }
}
```

Perbedaannya while mengecek kondisi terlebih dahulu, sehingga tidak menjamin bahwa blok kode akan dieksekusi jika kondisi awalnya salah, sebaliknya, do...while menjamin bahwa blok kode akan dieksekusi setidaknya sekali, sehingga sering kali digunakan ketika kita ingin memastikan bahwa tindakan tertentu terjadi minimal satu kali, terlepas dari kondisi awal.

2.3

```
import java.util.Scanner;

public class HitungRataRata {
    public static void main(String[] args) {
        int idPelajaran = 0;
        double nilaiPelajaran;
        double nilaiRataRata = 0;
        int jumlahPelajaran;

        Scanner scanner = new Scanner(System.in);
        System.out.print("Masukkan jumlah pelajaran: ");
        jumlahPelajaran = scanner.nextInt();

        while (idPelajaran < jumlahPelajaran) {
            System.out.print("Masukkan nilai pelajaran ke-" + (idPelajaran + 1) + ": ");
            nilaiPelajaran = scanner.nextDouble();
            nilaiRataRata = (nilaiPelajaran + nilaiRataRata) / (idPelajaran + 1);
            idPelajaran++;
        }

        System.out.println("Nilai rata-rata: " + nilaiRataRata);

        scanner.close();
    }
}
```

2.4 Rancang diagram flowchart dari Latihan 2.1, Latihan 2.2, dan Latihan 2.3!

a) Flowchart 2.1

1. Mulai
2. Inisialisasi idPelajaran ke 0
3. Inisialisasi nilaiRataRata ke 0
4. Minta pengguna untuk memasukkan jumlahPelajaran
5. Jika idPelajaran < jumlahPelajaran?
6. Jika Ya:
7. Minta pengguna memasukkan nilaiPelajaran
8. Hitung nilaiRataRata = (nilaiRataRata * idPelajaran + nilaiPelajaran) / (idPelajaran + 1)
9. idPelajaran++
10. Kembali ke langkah 5
11. Jika Tidak:
12. Tampilkan nilaiRataRata
13. Selesai

b) Flowchart 2.3

1. Mulai
2. Minta pengguna untuk memasukkan N
3. Inisialisasi i ke 1
4. Jika i <= N?

   Ya:

5. Jika i genap?

Ya: Tampilkan i

Tidak: Lanjutkan

6. i++

   Kembali ke langkah 4

   Tidak:

7. Selesai

c) Flowchart 3.1

1. Mulai
2. Inisialisasi a ke 0, b ke 1, dan jumlah ke N
3. Jika jumlah > 0?

   Ya:

4. Tampilkan a
5. c = a + b
6. a = b
7. b = c
8. Kurangi jumlah 1

   Kembali ke langkah 3

   Tidak:

9. Selesai

Kesimpulan

Kesimpulannya adalah saya telah mengeksplorasi berbagai konsep dasar dalam pemrograman Java, termasuk pernyataan while, do...while, dan penggunaan loop untuk menghitung rata-rata nilai pelajaran. saya juga mempelajari bagaimana kontrol alur dalam loop mempengaruhi output program, serta bagaimana pseudocode dapat diimplementasikan dalam bentuk kode Java yang jelas dan terstruktur.

Refleksi

Mempelajari tentang cara kerja struktur kontrol dalam pemrograman, yang penting untuk mengembangkan logika pemrograman yang efisien. Penggunaan flowchart untuk merancang algoritma juga memperkuat pemahaman tentang alur program dan membantu memvisualisasikan proses yang kompleks secara lebih sederhana.