

### Template Lembar Kerja Individu dan Kelompok

Nama & NPM	Topik:	Tanggal:
Yeni kusherawati G1F024013	FOR dan WHILE JAVA	08 oktober 2024

#### [Nomor 1] Identifikasi Masalah:

1) Uraikan permasalahan dan variabel

1.1. Analisa tujuan penulisan kata kunci `continue` dan `break` pada Contoh 1!

Buat perubahan nilai angka pada variabel di

//Ubah 1 menjadi `for (int y = 0; y <= 15; y++) { lalu running, periksa hasilnya`

//Ubah 2 menjadi `if (y % 2 == 0) lalu running, periksa hasilnya`

//Ubah 3 menjadi `else if (y == 9) lalu running, periksa hasilnya`

Analisa dampaknya perubahan ini terhadap luaran setelah running!

Jawab:

Continue: Melewatkan angka ganjil ( tidak mencetak)

Break: Menghentikan loop saat y sama dengan b

Dampak perubahannya adalah

ubah 1: `for (int y = 0; y <= 15; y++)` output: 0, 2, 4, 6 ( masih sama )

ubah 2: `if (y % 2 == 0)` output angka ganjil dari 1 hingga 15: 1, 3, 5, 7, 9, 11, 13, 15.

ubah 3: `else if (y == 9)` output angka ganjilnya hingga 1, 3, 5, 7.

Jadi kesimpulannya adalah perubahan mengubah output dari angka genap menjadi angka ganjil dan menghentikan loop lebih awal.

1.2. Buat perubahan kode pada Contoh 2 di baris //Ubah1 menjadi

a. `continue` pertama; lalu running, periksa hasilnya

b. `break` pertama; lalu running, periksa hasilnya

c. `continue` kedua; lalu running, periksa hasilnya

Analisa perbedaan perubahan kode pada Ubah 1 untuk setiap poin (a), (b), dan (c)!

Jawab:

Ketika `i==2`, `continue` kedua akan melewati iterasi saat ini dalam loop `j` dan langsung melanjutkan ke iterasi berikutnya. Pada `i=2`, `j` tetap mencetak 1 dan 2, sehingga outputnya sama dengan perubahan a. Jadi intinya `continue` pertama melanjutkan ke iterasi berikutnya dari loop luar (`i`), mencetak seluruh `j` untuk `i= 2`.

`Break` pertama menghentikan seluruh loop (`i`), hanya mencetak hasil untuk `i= 1`

`continue` kedua memungkinkan loop `j` menyelesaikan seluruh iterasi untuk `i= 2`, hasilnya sama dengan `continue` pertama.

1.3. Cermati kode contoh 3. Apabila ingin menghasilkan luaran berikut:

Luaran:

Masukan Input: 7

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

jawab: Logika looping

Loop pertama (`for(int t= tinggi; t>=1;t--)`): loop ini seharusnya mengontrol jumlah baris yang akan dicetak, yang seharusnya mulai dari tinggi hingga 1.

Loop kedua (`for(int s= tinggi; s>= t;s--)`) loop ini harus mencetak bintang (\*). Kondisinya menyebabkan hasil yang salah karena menampilkan bintang berdasarkan perbandingan `s` dengan `t`.

Perubahan yang diperlukan adalah ubah kondisi pada loop kedua agar mencetak

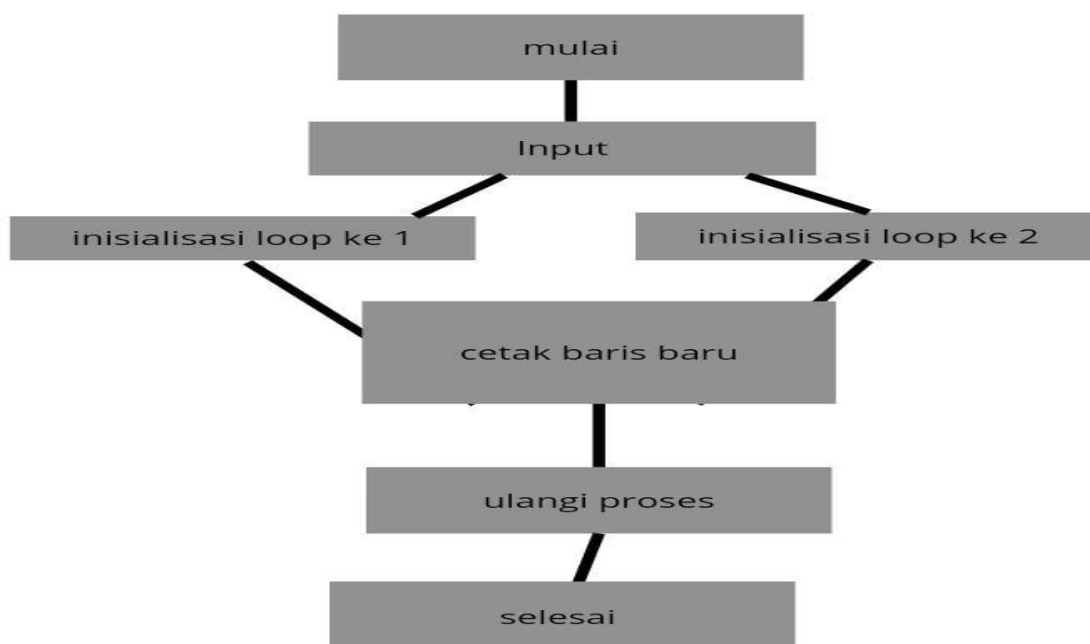
tinggi- $t + 1$  bintang, sehingga setiap baris akan memiliki jumlah bintang yang berkurang sesuai dengan barisnya.

1.4. Analisa diagram flowchart dari Latihan 1.2 dan 1.3!

1.2



1.3



2).Rincikan sumber informasi yang relevan (buku / webpage)

Video Materi 1 tentang FOR – <https://www.youtube.com/watch?v=Ij9qLLblxEU>

### [Nomor 1] Analisis dan Argumentasi

1) Uraikan rancangan solusi yang diusulkan.

Rancangan solusi yang diusulkan untuk semua masalah di atas berfokus pada penguasaan penggunaan struktur kontrol dalam pemrograman Java dan kemampuan untuk memodifikasi dan menganalisis kode untuk mencapai hasil yang diinginkan. Pendekatan ini berfokus pada pemahaman langsung dengan bereksperimen dengan kode.

2) Analisis solusi, kaitkan dengan permasalahan.

Analisis solusi menunjukkan bahwa pendekatan yang diusulkan efektif dalam mengatasi permasalahan yang ada. Dengan berfokus pada eksplorasi dan pemahaman mendalam tentang struktur kontrol, pengguna tidak hanya dapat memecahkan masalah kesalahan tetapi juga memperkuat keterampilan pemrograman mereka secara keseluruhan.

Solusi ini memberikan dasar yang kuat untuk memahami konsep pemrograman Java tingkat lanjut.

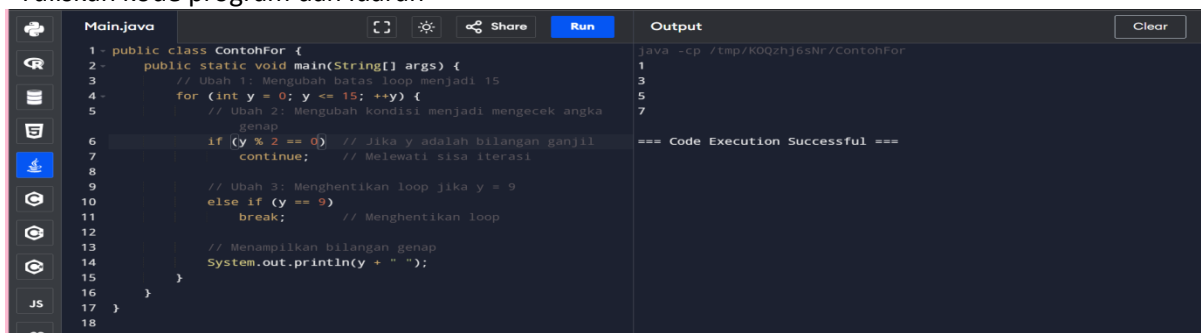
### [Nomor 1] Penyusunan Algoritma dan Kode Program

1.1 Rancang desain solusi atau algoritma

Algoritma

1. Loop dari 0 hingga 15
2. Lewati bilangan genap
3. Hentikan loop jika y mencapai 9
4. Cetak bilangan ganjil yang ditemukan
5. Hasil keluaran yang dicetak 1, 3, 5, 7.

Tuliskan kode program dan luaran



```
1 - public class ContohFor {
2 -     public static void main(String[] args) {
3         // Ubah 1: Mengubah batas loop menjadi 15
4         for (int y = 0; y <= 15; ++y) {
5             // Ubah 2: Mengubah kondisi menjadi mengecek angka
6             // genap
7             if (y % 2 == 0) // Jika y adalah bilangan ganjil
8                 continue; // Melewati sisa iterasi
9
10            // Ubah 3: Menghentikan loop jika y = 9
11            else if (y == 9)
12                break; // Menghentikan loop
13
14            // Menampilkan bilangan ganap
15            System.out.println(y + " ");
16        }
17    }
18 }
```

Output

```
java -cp /tmp/KOQzhj6sNr/ContohFor
1
3
5
7
=== Code Execution Successful ===
```

Berikan komentar pada program ini

Loop dimulai dari y= 0 hingga y=15, cek genap jika y genap iterasi di lewati dengan continue, cek nilai 9 jika y=9, loop dihentikan dengan break, dan outputnya akan mencetak bilangan ganjil sebelum y = 9.

1.2 Rancang desain solusi atau algoritma

Algoritma

1. Inisialisasi program ( buat kelas forBersarang)
2. Gunakan break pada loop bersarang
3. Gunkan continue pada loop bersarang
4. Selesai.

Tuliskan kode program dan luaran

```

Main.java
1 public class ForBersarang {
2     public static void main(String[] args) {
3         // Bagian 1: Menggunakan 'break' pada loop kedua
4         System.out.println("Menggunakan 'break':");
5         pertama:
6         for (int i = 1; i < 5; i++) {
7             kedua:
8             for (int j = 1; j < 3; j++) {
9                 System.out.println("i = " + i + "; j = " + j);
10                if (i == 2)
11                    break kedua; // Ubah 1
12            }
13        }
14
15        System.out.println("\nMenggunakan 'continue' pertama:");
16    }
17    // Bagian 2: Menggunakan 'continue' pada loop kedua
18    pertama:
19    for (int i = 1; i < 5; i++) {
20        kedua:

```

```

Output
java -cp /tmp/1BMS2Rwpgu/ForBersarang
Menggunakan 'break':
i = 1; j = 1
i = 1; j = 2
i = 2; j = 1
i = 2; j = 2
i = 3; j = 1
i = 3; j = 2
i = 4; j = 1
i = 4; j = 2
Menggunakan 'continue' pertama:
i = 1; j = 1
i = 1; j = 2
i = 2; j = 1
i = 2; j = 2
i = 3; j = 1
i = 3; j = 2
i = 4; j = 1
i = 4; j = 2

```

Berikan komentar pada program ini

Kode ini menunjukkan perbedaan pengguna break untuk menghentikan loop, dan continue untuk melewati iterasi tanpa menghentikan loop.

### 1.3 Rancang desain solusi atau algoritma

Algoritma

1. Mulai
2. Buat instance dari scanner untuk mengambil input dari pengguna
3. Tampilkan pesan untuk meminta input dari pengguna
4. Terima input
5. Lakukan perulangan
6. Ulangi langkah langkah yang telah dilakukan sebelumnya
7. Selesai

Tuliskan kode program dan luaran

```

Main.java
1 import java.util.Scanner;
2
3 public class ForBersarang {
4     public static void main(String[] args) {
5         // Instance Input Scanner
6         Scanner input = new Scanner(System.in);
7         System.out.print("Masukan Input: ");
8
9         int tinggi = input.nextInt(); // Mendapatkan Input Dari User
10
11         // Loop untuk mencetak baris bintang
12         for (int t = tinggi; t >= 1; t--) {
13             // Menghitung Jumlah Bintang per Baris
14             for (int s = 1; s <= t; s++) {
15                 System.out.print("*");
16             }
17             System.out.println(); // Membuat Baris Baru
18         }
19     }

```

```

Output
java -cp /tmp/nwLZ4ixPC6/ForBersarang
Masukan Input: 7
*****
*****
****
***
**
*
=== Code Execution Successful ===

```

Berilah komentar pada program ini

Program meminta input berupa angka yang menentukan tinggi segitiga bintang terbalik. Setelah mencetak bintang di setiap baris, pindah ke baris baru untuk mulai mencetak bintang di baris berikutnya dengan jumlah yang lebih sedikit. Hasil akhirnya adalah segitiga terbalik dari bintang sesuai input.

<b>[Nomor 1] Kesimpulan</b>
1) Analisa Tingkatkan pemahaman tentang cara melanjutkan dan menjeda pekerjaan. Perubahan kondisi dan hasil keluaran menunjukkan bagaimana struktur kendali mempengaruhi alur program. Menjelaskan cara mencetak pola menggunakan loop bersarang. Perubahan pada logika pengulangan akan memberi tahu bagaimana karakter dihitung dan dicetak sesuai kebutuhan. Menekankan pentingnya memahami logika dalam loop untuk menggunakan kontrol iterasi yang tepat untuk menghasilkan keluaran yang diinginkan.

<b>Nama &amp; NPM</b>	<b>Topik:</b>	<b>Tanggal:</b>
<b>Yeni kusherawati G1F024013</b>	<b>FOR dan WHILE JAVA</b>	<b>08 oktober 2024</b>

**[Nomor 2] Identifikasi Masalah:**

1. Uraikan permasalahan dan variabel

2.1. Buat perubahan nilai angka pada variabel di Contoh 4

//Ubah 1 menjadi continue; lalu running, periksa hasilnya

Analisa dampaknya perubahan terhadap luaran setelah running dan uraikan kegunaan break dan continue!

Jawab:

Break menghentikan eksekusi loop sepenuhnya. Berguna ketika ingin keluar dari loop berdasarkan kondisi tertentu, seperti menemukan nilai yang dicari.

Continue melewati iterasi saat ini dan melanjutkan ke iterasi berikutnya. Ini berguna untuk menghindari eksekusi kode tertentu terpenuhi, misalnya mengabaikan nilai yang tidak diinginkan. Jadi intinya adalah perubahan dari break ke continue secara signifikan mengubah hasil keluaran program. Memahami cara kerja kedua pernyataan ini penting untuk pengendalian aliran dalam pemrograman.

2.2 Buat perubahan nilai angka pada variabel di Contoh 5

//Ubah2 menjadi `if (count % 5 == 0)` lalu running, periksa hasilnya

Analisa dampaknya perubahan terhadap luaran setelah running dan uraikan kegunaan % untuk angka yang berbeda pada perintah tersebut!

Jawab:

Penggunaan operator modulus (%) memungkinkan pemrogram untuk memeriksa apakah suatu angka adalah kelipatan dari angka lain. Dengan mengganti angka dalam kondisi modulus, program dapat dengan mudah disesuaikan untuk mencetak kelipatan yang berbeda, yang berguna dalam banyak konteks pemrograman untuk seleksi dan perhitungan.

2.3 Buat perubahan nilai angka pada variabel di

//Ubah1 menjadi `while (count < 0) {` lalu running, periksa hasilnya

Ubahlah baris kode `while` pada Contoh 5 menjadi `do ... while` dengan persyaratan yang sama `while (count < 0)`. Bandingkan hasil luaran antara menggunakan `while` dan `do ... while`!

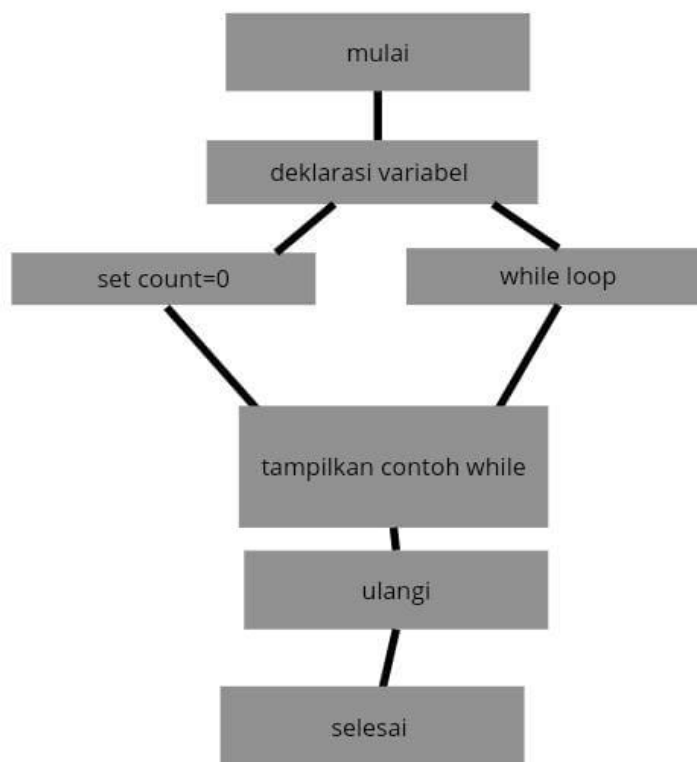
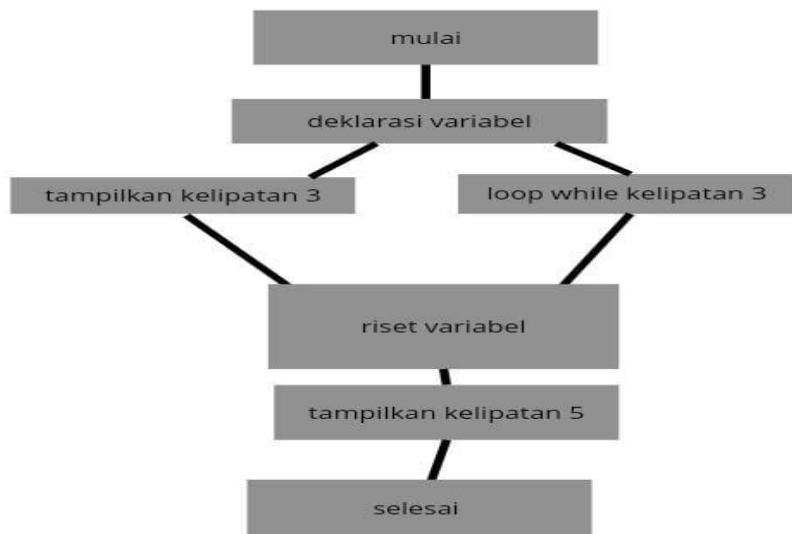
Jawab:

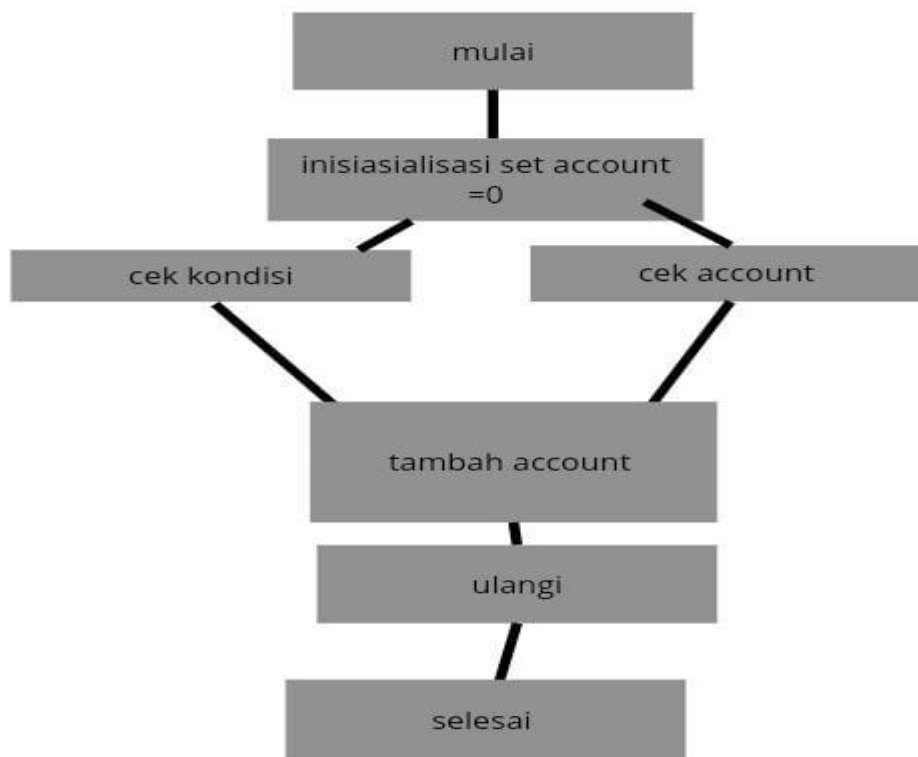
While: Loop tidak di eksekusi sama sekali jika kondisi awal tidak terpenuhi

do..while: Bagian dalam `do` dieksekusi minimal sekali, terlepas dari apakah kondisi terpenuhi atau tidak. Dalam kasus ini, tidak ada perbedaan output karena kondisi tidak terpenuhi, tetapi penting untuk diingat bahwa `do..while` menjamin eksekusi setidaknya satu kali.

2.4. Analisa diagram flowchart dari Latihan 2.1, Contoh 5, dan Latihan 2.3!

2.1





5.

2. Rincikan sumber informasi yang relevan (buku / webpage)  
Video Materi 2 tentang WHILE – <https://www.youtube.com/watch?v=ORA4JyJMFss>

#### [Nomor 2] Analisis dan Argumentasi

- 1) Uraikan rancangan solusi yang diusulkan.  
Rancangan solusi yang diusulkan bertujuan untuk meningkatkan pemahaman tentang kontrol aliran, loop, dan bagaimana struktur logika dalam kode dapat mempengaruhi hasil output. Dengan pendekatan yang interaktif dan praktis, pengguna dapat lebih mudah menguasai konsep-konsep penting dalam pemrograman.
- 2) Analisis solusi, kaitkan dengan permasalahan  
Analisis solusi menunjukkan bahwa setiap elemen dalam soal berkaitan langsung dengan permasalahan dasar dalam pemrograman, yaitu pengendalian aliran dan pengolahan data. Dengan memahami dan menerapkan konsep-konsep ini, pengguna dapat menulis kode yang lebih efisien, fleksibel, dan mudah dipahami.

#### [Nomor 2] Penyusunan Algoritma dan Kode Program

2.1 Rancang desain solusi atau algoritma

Algoritma

1. Inisialisasi
2. Mulai loop
3. Tampilkan nilai i=1
4. Selesai (program berakhir setelah loop)

Tuliskan kode program dan luaran



The screenshot shows a Java IDE with a file named 'Main.java'. The code defines a class 'ContohWhile' with a 'main' method. It demonstrates two types of while loops: one using 'break' to exit a loop and another using 'continue' to skip an iteration. The output window shows the execution results, including the numbers 1 through 3 for the first loop and 1 through 6 for the second loop, followed by a success message.

```

1- public class ContohWhile {
2-     public static void main(String[] args) {
3-         // Contoh dengan break
4-         System.out.println("Contoh dengan break:");
5-         int i = 1;
6-         while (i <= 6) {
7-             System.out.println(i);
8-             i++;
9-             if (i == 4) {
10-                 break; // menghentikan loop
11-             }
12-         }
13-
14-         // Reset nilai i untuk contoh kedua
15-         i = 1;
16-
17-         // Contoh dengan continue
18-         System.out.println("\nContoh dengan continue:");
19-         while (i <= 6) {
20-             if (i == 4) {

```

```

java -cp /tmp/MxB8Xyh3n0/ContohWhile
Contoh dengan break:
1
2
3

Contoh dengan continue:
1
2
3
5
6

=== Code Execution Successful ===

```

## 2.2 Rancang desain solusi atau algoritma

### Algoritma

1. Inisialisasi
2. Mulai loop
3. Cek kondisi
4. Increment
5. Selesai

Tuliskan kode program dan luaran

The screenshot shows a Java IDE with a file named 'Main.java'. The code defines a class 'WhileBersarang' with a 'main' method. It demonstrates a while loop with an if statement inside, printing multiples of 3 and 5. The output window shows the execution results, including the numbers 0, 3, 6, 9, 12, 15, 18 for the first loop and 0, 5, 10, 15 for the second loop, followed by a success message.

```

1- public class WhileBersarang {
2-     public static void main(String[] args) {
3-         // Contoh dengan if (count % 3 == 0)
4-         System.out.println("Kelipatan 3:");
5-         int count = 0;
6-         while (count < 20) {
7-             if (count % 3 == 0) {
8-                 System.out.println(count);
9-             }
10-             count++;
11-         }
12-
13-         // Reset nilai count untuk contoh kedua
14-         count = 0;
15-
16-         // Contoh dengan if (count % 5 == 0)
17-         System.out.println("\nKelipatan 5:");
18-         while (count < 20) {
19-             if (count % 5 == 0) {
20-                 System.out.println(count);

```

```

java -cp /tmp/I3bk7V6hjZ/WhileBersarang
Kelipatan 3:
0
3
6
9
12
15
18

Kelipatan 5:
0
5
10
15

=== Code Execution Successful ===

```

## 2.3

Tuliskan kode program dan luaran

The screenshot shows a Java IDE with a file named 'Main.java'. The code defines a class 'WhileBersarang' with a 'main' method. It demonstrates a while loop and a do-while loop. The output window shows the execution results, including the numbers 0, 3, 6, 9, 12, 15, 18 for the first loop and 0 for the second loop, followed by a success message.

```

1- public class WhileBersarang {
2-     public static void main(String[] args) {
3-         int count;
4-
5-         // Contoh dengan while
6-         System.out.println("Contoh dengan while (count < 0):");
7-         count = 0; //ubah1
8-         while (count < 0) {
9-             if (count % 3 == 0) //ubah2
10-                 System.out.println(count);
11-             count++;
12-         }
13-
14-         // Contoh dengan do...while
15-         System.out.println("\nContoh dengan do...while (count < 0):");
16-         count = 0; //reset nilai count
17-         do {
18-             if (count % 3 == 0) //ubah2
19-                 System.out.println(count);

```

```

java -cp /tmp/ooTpaTtyuP/WhileBersarang
Contoh dengan while (count < 0):
0

Contoh dengan do...while (count < 0):
0

=== Code Execution Successful ===

```

<b>[Nomor 2] Kesimpulan</b>
3) Analisa Tingkatkan pemahaman tentang cara melanjutkan dan menjeda pekerjaan. Perubahan kondisi dan hasil keluaran menunjukkan bagaimana struktur kendali mempengaruhi alur program. Menjelaskan cara mencetak pola menggunakan loop bersarang. Perubahan pada logika pengulangan akan memberi tahu bagaimana karakter dihitung dan dicetak sesuai kebutuhan. Menekankan pentingnya memahami logika dalam loop untuk menggunakan kontrol iterasi yang tepat untuk menghasilkan keluaran yang diinginkan.