Modul 6 Kelas Untuk Berbagai Type Data

Tugas Pendahuluan:

1. Jelaskan apa itu Class Number dalam bahasa pemrograman Java.
2. Jelaskan apa itu Class Char dalam bahasa pemrograman Java.
3. Jelaskan apa itu Class String dalam bahasa pemrograman Java.
4. Jelaskan apa itu Class Array dalam bahasa pemrograman Java.
5. Jelaskan apa itu Class Date and Time dalam bahasa pemrograman Java.

Materi

1. Numbers Class

Dalam bahasa pemrograman Java, Numbers Class adalah kumpulan dari berbagai tipe data primitif angka yang dikemas dalam bentuk kelas wrapper (wrapper class). Wrapper class adalah kelas yang menyediakan objek yang mengenkapsulasi tipe data primitif sehingga memungkinkan untuk melakukan operasi tambahan dan mengakses metode yang relevan dengan tipe data tersebut.

Tipe data primitif angka seperti int, double, float, long, byte, short, dan char memiliki masing-masing wrapper class yang sesuai:

1) Integer: Wrapper class untuk tipe data int.
2) Double: Wrapper class untuk tipe data double.
3) Float: Wrapper class untuk tipe data float.
4) Long: Wrapper class untuk tipe data long.
5) Byte: Wrapper class untuk tipe data byte.
6) Short: Wrapper class untuk tipe data short.
7) Character: Wrapper class untuk tipe data char.

Numbers Class menyediakan beberapa metode statis yang memungkinkan kita untuk melakukan operasi matematika atau konversi dengan tipe data angka. Beberapa contoh metode yang disediakan adalah:

1) **parseInt(String s)**: Mengubah string menjadi tipe data int.
2) **parseDouble(String s)**: Mengubah string menjadi tipe data double.
3) **valueOf(int i)**: Mengubah tipe data int menjadi objek Integer.
4) **valueOf(double d)**: Mengubah tipe data double menjadi objek Double.

Contoh penggunaan:

```
1. public class NumbersClassExample {
2.     public static void main(String[] args) {
3.         String numStr = "123";
4.         int numInt = Integer.parseInt(numStr);
5.         System.out.println("numInt: " + numInt);
6.
7.         double numDouble = Double.parseDouble(numStr);
8.         System.out.println("numDouble: " + numDouble);
9.
10.        int intValue = 42;
11.        Integer intObject = Integer.valueOf(intValue);
12.        System.out.println("intObject: " + intObject);
13.    }
14. }
15.
```

Output :

```
numInt: 123
numDouble: 123.0
intObject: 42
```

Dalam contoh di atas, kita menggunakan metode **parseInt** dan **parseDouble** dari Numbers Class untuk mengubah string menjadi tipe data int dan double. Selain itu, kita menggunakan metode **valueOf** untuk mengubah tipe data int menjadi objek Integer. Penggunaan Numbers Class ini sangat berguna ketika kita perlu bekerja dengan operasi matematika atau melakukan konversi tipe data angka di Java.

2. Character Class

Dalam bahasa pemrograman Java, Character Class adalah sebuah kelas wrapper yang menyediakan beberapa metode untuk mengakses dan memanipulasi karakter (tipe data char). Karakter dalam Java direpresentasikan oleh tipe data primitif char, dan Character Class mengemasnya ke dalam sebuah objek sehingga memungkinkan kita untuk melakukan operasi tambahan yang relevan dengan karakter.

1) Character Class menyediakan metode-metode yang berguna untuk mengakses dan memanipulasi karakter. Beberapa di antaranya adalah:
2) isLetter(char ch): Memeriksa apakah karakter merupakan huruf (a-z atau A-Z).
3) isDigit(char ch): Memeriksa apakah karakter merupakan digit angka (0-9).
4) isWhitespace(char ch): Memeriksa apakah karakter merupakan spasi, tab, atau karakter whitespace lainnya.
5) toUpperCase(char ch): Mengubah karakter menjadi huruf kapital jika karakter tersebut adalah huruf.
6) toLowerCase(char ch): Mengubah karakter menjadi huruf kecil jika karakter tersebut adalah huruf.
7) toString(char ch): Mengubah karakter menjadi tipe data String.

Contoh penggunaan:

```
1. public class CharacterClassExample {
2.     public static void main(String[] args) {
3.         char ch = 'A';
4.
5.         if (Character.isLetter(ch)) {
6.             System.out.println(ch + " adalah huruf.");
7.         } else {
8.             System.out.println(ch + " bukan huruf.");
9.         }
10.
11.        if (Character.isDigit(ch)) {
12.            System.out.println(ch + " adalah digit angka.");
13.        } else {
14.            System.out.println(ch + " bukan digit angka.");
15.        }
16.
17.        if (Character.isWhitespace(ch)) {
18.            System.out.println(ch + " adalah spasi atau karakter whitespace.");
19.        } else {
20.            System.out.println(ch + " bukan spasi atau karakter whitespace.");
21.        }
22.
23.        char lowercaseCh = Character.toLowerCase(ch);
24.        System.out.println("Huruf " + ch + " diubah menjadi " + lowercaseCh);
25.
26.        String chString = Character.toString(ch);
27.        System.out.println("Karakter " + ch + " dalam bentuk String: " + chString);
28.    }
29. }
30.
```

Output :

```
A adalah huruf.
A bukan digit angka.
A bukan spasi atau karakter whitespace.
Huruf A diubah menjadi a
Karakter A dalam bentuk String: A
```

Dalam contoh di atas, kita menggunakan metode-metode dari Character Class untuk memeriksa apakah karakter adalah huruf, digit angka, atau spasi. Kita juga menggunakan metode toLowerCase untuk mengubah karakter menjadi huruf kecil, dan metode toString untuk mengubah karakter menjadi tipe data String. Character Class ini sangat berguna ketika kita perlu bekerja dengan karakter atau melakukan validasi terhadap input yang mengandung karakter tertentu.

3. Strings Class

Dalam bahasa pemrograman Java, "Strings Class" tidak ada. Yang ada adalah "String Class" (kata "String" dalam bentuk singular), yang merupakan salah satu kelas paling penting dan sering digunakan

dalam Java. String Class merupakan kelas yang membungkus objek dari tipe data String, yang merupakan representasi dari urutan karakter (teks) dalam Java.

String Class menyediakan berbagai metode yang memungkinkan kita untuk melakukan berbagai operasi pada objek String, seperti manipulasi, pencarian, pemotongan, dan banyak lagi. Beberapa metode yang sering digunakan dalam String Class antara lain:

1) length(): Mengembalikan panjang (jumlah karakter) dari sebuah String.
2) charAt(int index): Mengembalikan karakter pada indeks tertentu dari sebuah String.
3) concat(String str): Menggabungkan dua String menjadi satu String baru.
4) equals(Object obj): Membandingkan apakah dua String memiliki nilai yang sama.
5) substring(int beginIndex): Mengambil potongan dari sebuah String mulai dari indeks tertentu hingga akhir String.
6) substring(int beginIndex, int endIndex): Mengambil potongan dari sebuah String antara indeks tertentu hingga indeks sebelum endIndex.
7) indexOf(String str): Mencari indeks pertama dari substring tertentu dalam sebuah String.

Contoh penggunaan:

```
 1. public class StringsClassExample {
 2.     public static void main(String[] args) {
 3.         String str1 = "Hello";
 4.         String str2 = "World";
 5.
 6.         // Menggunakan metode length()
 7.         int lengthStr1 = str1.length();
 8.         System.out.println("Panjang str1: " + lengthStr1);
 9.
10.         // Menggunakan metode charAt()
11.         char charAtIndex = str1.charAt(0);
12.         System.out.println("Karakter pada indeks 0: " + charAtIndex);
13.
14.         // Menggunakan metode concat()
15.         String concatenatedStr = str1.concat(str2);
16.         System.out.println("Hasil penggabungan: " + concatenatedStr);
17.
18.         // Menggunakan metode equals()
19.         boolean isEqual = str1.equals(str2);
20.         System.out.println("Apakah str1 sama dengan str2? " + isEqual);
21.
22.         // Menggunakan metode substring()
23.         String substringStr = str1.substring(1, 4);
24.         System.out.println("Potongan str1: " + substringStr);
25.
26.         // Menggunakan metode indexOf()
27.         int index = str1.indexOf("l");
28.         System.out.println("Indeks pertama karakter 'l' pada str1: " + index);
29.     }
30. }
31.
```

Output :

```
Panjang str1: 5
Karakter pada indeks 0: H
Hasil penggabungan: HelloWorld
Apakah str1 sama dengan str2? false
Potongan str1: ell
Indeks pertama karakter 'l' pada str1: 2
```

Dalam contoh di atas, kita menggunakan beberapa metode dari String Class untuk melakukan manipulasi dan pencarian pada objek String. String Class ini sangat sering digunakan dalam pemrograman Java karena representasi teks yang sering ditemui dalam aplikasi dan pengolahan data.

4. Array

Dalam bahasa pemrograman Java, "Arrays" adalah struktur data yang digunakan untuk menyimpan kumpulan elemen dengan tipe data yang sama. Elemen-elemen ini diakses melalui indeks, di mana indeks pertama memiliki nilai 0, indeks kedua memiliki nilai 1, dan seterusnya. Array digunakan untuk menyimpan data berulang atau data yang memerlukan akses dengan indeks, seperti data numerik, data dalam urutan tertentu, atau kumpulan objek yang seragam.

Untuk membuat dan menggunakan array dalam Java, berikut adalah langkah-langkah yang umum dilakukan:

1) Deklarasi array: Menentukan tipe data elemen array dan menyatakan nama variabel array.
2) Inisialisasi array: Mengalokasikan ruang memori untuk array dan menentukan ukuran array. Array akan berisi nilai default sesuai dengan tipe datanya (misalnya 0 untuk int, null untuk objek).
3) Akses elemen array: Mengakses nilai elemen array menggunakan indeks.

Contoh pembuatan dan penggunaan array dalam Java:

```
 1. public class ArraysExample {
 2.     public static void main(String[] args) {
 3.         // Contoh deklarasi dan inisialisasi array integer
 4.         int[] numbers = new int[5]; // Array dengan ukuran 5
 5.         numbers[0] = 10;
 6.         numbers[1] = 20;
 7.         numbers[2] = 30;
 8.         numbers[3] = 40;
 9.         numbers[4] = 50;
10.
11.         // Contoh akses elemen array
12.         System.out.println("Elemen ke-0: " + numbers[0]);
13.         System.out.println("Elemen ke-2: " + numbers[2]);
14.
15.         // Contoh deklarasi dan inisialisasi array string
16.         String[] names = {"Alice", "Bob", "Charlie", "David"};
17.
18.         // Contoh akses elemen array
19.         System.out.println("Nama pertama: " + names[0]);
20.         System.out.println("Nama terakhir: " + names[3]);
21.
```

```
22.         // Contoh looping untuk mengakses semua elemen array
23.         System.out.println("Semua elemen array numbers:");
24.         for (int i = 0; i < numbers.length; i++) {
25.             System.out.println(numbers[i]);
26.         }
27.
28.         System.out.println("Semua elemen array names:");
29.         for (String name : names) {
30.             System.out.println(name);
31.         }
32.     }
33. }
34.
```

Output :

```
Elemen ke-0: 10
Elemen ke-2: 30
Nama pertama: Alice
Nama terakhir: David
Semua elemen array numbers:
10
20
30
40
50
Semua elemen array names:
Alice
Bob
Charlie
David
```

Dalam contoh di atas, kita membuat dua array, yaitu array "numbers" dengan ukuran 5 untuk menyimpan data integer, dan array "names" yang langsung diinisialisasi dengan nilai string. Kemudian kita mengakses elemen-elemen array menggunakan indeks dan melakukan iterasi (looping) untuk mengakses semua elemen array menggunakan for-each loop. Array sangat berguna ketika kita perlu menyimpan data dalam jumlah besar dan memprosesnya secara efisien.

5. Date and Time

Dalam bahasa pemrograman Java, "Date and Time" adalah tentang cara memanipulasi dan bekerja dengan tanggal, waktu, dan durasi. Sejak Java 8, Java menyediakan package java.time yang menyediakan banyak kelas dan metode untuk mengelola date and time dengan lebih baik dibandingkan dengan API lama yang terdapat di java.util. Beberapa kelas penting dalam package java.time antara lain:

1) LocalDate: Mengrepresentasikan tanggal (tahun, bulan, dan hari) tanpa waktu atau zona waktu.
2) LocalTime: Mengrepresentasikan waktu (jam, menit, detik, dan nanodetik) tanpa tanggal atau zona waktu.
3) LocalDateTime: Mengrepresentasikan tanggal dan waktu tanpa zona waktu.
4) ZonedDateTime: Mengrepresentasikan tanggal, waktu, dan zona waktu.

5) Duration: Mengrepresentasikan selisih waktu antara dua titik waktu dalam satuan detik atau nanodetik.
6) Period: Mengrepresentasikan selisih tanggal antara dua titik waktu dalam satuan tahun, bulan, atau hari.

Contoh penggunaan:

```
 1. import java.time.LocalDate;
 2. import java.time.LocalTime;
 3. import java.time.LocalDateTime;
 4. import java.time.ZonedDateTime;
 5. import java.time.Duration;
 6. import java.time.Period;
 7.
 8. public class DateTimeExample {
 9.     public static void main(String[] args) {
10.         // LocalDate: Mengambil tanggal saat ini
11.         LocalDate currentDate = LocalDate.now();
12.         System.out.println("Tanggal saat ini: " + currentDate);
13.
14.         // LocalTime: Mengambil waktu saat ini
15.         LocalTime currentTime = LocalTime.now();
16.         System.out.println("Waktu saat ini: " + currentTime);
17.
18.         // LocalDateTime: Menggabungkan tanggal dan waktu saat ini
19.         LocalDateTime currentDateTime = LocalDateTime.now();
20.         System.out.println("Tanggal dan waktu saat ini: " + currentDateTime);
21.
22.         // ZonedDateTime: Mengambil tanggal, waktu, dan zona waktu saat ini
23.         ZonedDateTime currentZonedDateTime = ZonedDateTime.now();
24.         System.out.println("Tanggal, waktu, dan zona waktu saat ini: " + currentZonedDateTime);
25.
26.         // Duration: Menghitung selisih waktu antara dua titik waktu
27.         LocalDateTime startDateTime = LocalDateTime.of(2023, 7, 1, 0, 0);
28.         LocalDateTime endDateTime = LocalDateTime.of(2023, 7, 5, 12, 30);
29.         Duration duration = Duration.between(startDateTime, endDateTime);
30.         System.out.println("Selisih waktu: " + duration.toHours() + " jam");
31.
32.         // Period: Menghitung selisih tanggal antara dua titik waktu
33.         LocalDate startDate = LocalDate.of(2023, 7, 1);
34.         LocalDate endDate = LocalDate.of(2023, 7, 5);
35.         Period period = Period.between(startDate, endDate);
36.         System.out.println("Selisih tanggal: " + period.getDays() + " hari");
37.     }
38. }
39.
```

Output :

```
Tanggal saat ini: 2023-07-05
Waktu saat ini: 10:25:30.123456789
Tanggal dan waktu saat ini: 2023-07-05T10:25:30.123456789
Tanggal, waktu, dan zona waktu saat ini: 2023-07-05T10:25:30.123456789+07:00[Asia/Jakarta]
Selisih waktu: 12 jam
Selisih tanggal: 4 hari
```

Dalam contoh di atas, kita menggunakan beberapa kelas dari package java.time untuk mengelola date and time. Package ini memungkinkan kita untuk melakukan manipulasi, perhitungan, dan representasi tanggal, waktu, dan durasi dengan lebih mudah dan akurat. Sebagai catatan, perhatikan bahwa representasi tanggal dan waktu dalam package java.time adalah immutable, yang artinya setelah objek dibuat, nilainya tidak dapat diubah.

Latihan

Berikut adalah beberapa latihan untuk memperbaiki kesalahan program pada masing-masing class, yaitu Class Number, Char, String, Array, dan Date and Time. Pada setiap latihan, Anda akan diberikan sebuah program yang memiliki kesalahan, dan tugas Anda adalah untuk mengidentifikasi dan memperbaiki kesalahan tersebut sehingga program dapat berjalan dengan benar.

1) Latihan Memperbaiki Kesalahan Program pada Class Number:

```
1. public class NumberExample {
2.     public static void main(String[] args) {
3.         int x = 10;
4.         int y = 5;
5.
6.         // Menggunakan method Integer.toString() untuk mengubah int menjadi String
7.         String xStr = Integer.toString(x);
8.         String yStr = Integer.toString(y);
9.
10.        // Menggabungkan dua String
11.        String result = xStr + yStr;
12.        System.out.println("Hasil penggabungan: " + result);
13.    }
14. }
15.
```

Program di atas memiliki kesalahan karena kita mencoba untuk menggabungkan dua nilai integer (x dan y) dengan menggunakan operator +. Namun, untuk menggabungkan dua nilai integer menjadi String, kita perlu mengubahnya terlebih dahulu menjadi String menggunakan metode Integer.toString().

2) Latihan Memperbaiki Kesalahan Program pada Class Char:

```
1. public class CharExample {
2.     public static void main(String[] args) {
3.         char ch = 'A';
4.
5.         // Menggunakan metode Character.isLetter() untuk memeriksa apakah karakter adalah huruf
6.         if (Character.isLetter(ch)) {
7.             System.out.println(ch + " adalah huruf.");
8.         } else {
9.             System.out.println(ch + " bukan huruf.");
10.        }
11.    }
12. }
13.
```

Program di atas memiliki kesalahan karena kita mencoba untuk mencetak karakter ch tanpa mengubahnya menjadi String terlebih dahulu. Karena ch adalah karakter, untuk mencetaknya kita perlu mengubahnya menjadi String dengan menambahkan tanda kutip tunggal ('') di sekitarnya.

3) Latihan Memperbaiki Kesalahan Program pada Class String:

```
 1. public class StringExample {
 2.     public static void main(String[] args) {
 3.         String str = "Hello World";
 4.
 5.         // Menggunakan metode String.substring() untuk memotong substring dari sebuah String
 6.         String subStr = str.substring(0, 5);
 7.         System.out.println("Substring: " + subStr);
 8.     }
 9. }
10.
```

Program di atas memiliki kesalahan karena indeks pada metode substring harus dimulai dari 0 dan kurang dari panjang String. Jadi, untuk mendapatkan substring dari indeks 0 hingga 4, kita perlu menggunakan str.substring(0, 5).

4) Latihan Memperbaiki Kesalahan Program pada Class Array:

```
 1. public class ArrayExample {
 2.     public static void main(String[] args) {
 3.         // Deklarasi dan inisialisasi array integer
 4.         int[] numbers = {1, 2, 3, 4, 5};
 5.
 6.         // Menggunakan loop untuk mengakses dan mencetak semua elemen array
 7.         System.out.println("Semua elemen array numbers:");
 8.         for (int i = 0; i < numbers.length; i++) {
 9.             System.out.println(numbers[i]);
10.         }
11.     }
12. }
13.
```

Program di atas memiliki kesalahan karena kita mencetak elemen array menggunakan indeks yang dimulai dari 1, padahal indeks array dimulai dari 0. Jadi, untuk mencetak semua elemen array, kita perlu menggunakan numbers[i].

5) Latihan Memperbaiki Kesalahan Program pada Class Date and Time:

```
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

public class DateTimeExample {
    public static void main(String[] args) {
        // Mengambil tanggal saat ini
        LocalDate currentDate = LocalDate.now();

        // Menggunakan DateTimeFormatter untuk mencetak tanggal dalam format yang sesuai
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy");
        String formattedDate = currentDate.format(formatter);
```

```
        System.out.println("Tanggal saat ini: " + formattedDate);
    }
}
```

Program di atas memiliki kesalahan karena kita mencoba untuk mencetak tanggal menggunakan currentDate langsung tanpa menggunakan DateTimeFormatter. Karena currentDate adalah objek dari LocalDate, kita perlu menggunakan DateTimeFormatter untuk memformat tanggal sesuai dengan format yang diinginkan.

Tugas Rumah

Pada tugas rumah ini, Anda diminta untuk membuat program Java yang menggunakan konsep dan fitur dari Class Number, Char, String, Array, dan Date and Time. Program yang akan Anda buat harus dapat mengakses, memanipulasi, atau menampilkan data menggunakan class-class tersebut. Berikut adalah deskripsi tugasnya:

1) Membuat Program Menggunakan Class Number:
   Buatlah program Java yang mengambil dua angka dari pengguna dan kemudian menampilkan hasil penjumlahan, pengurangan, perkalian, dan pembagian dari kedua angka tersebut.

2) Membuat Program Menggunakan Class Char:
   Buatlah program Java yang mengambil sebuah karakter (huruf) dari pengguna dan kemudian menampilkan apakah karakter tersebut adalah huruf besar (kapital) atau huruf kecil.

3) Membuat Program Menggunakan Class String:
   Buatlah program Java yang meminta pengguna untuk memasukkan sebuah kalimat atau kata, kemudian program tersebut akan memeriksa apakah kalimat tersebut adalah palindrom atau bukan. Sebuah kalimat atau kata dikatakan palindrom jika dibaca dari depan maupun dari belakang memiliki urutan karakter yang sama.

4) Membuat Program Menggunakan Class Array:
   Buatlah program Java yang mengambil beberapa angka dari pengguna (misalnya 5 angka), kemudian program tersebut akan menampilkan nilai terkecil dan terbesar dari angka-angka tersebut.

5) Membuat Program Menggunakan Class Date and Time:
   Buatlah program Java yang menampilkan tanggal dan waktu saat ini dalam format yang sesuai, lalu program tersebut akan menghitung dan menampilkan tanggal dan waktu setelah 100 hari dari saat ini.

Pastikan untuk memberikan petunjuk yang jelas kepada pengguna tentang apa yang diharapkan dari setiap program dan bagaimana cara penggunaan program tersebut. Selain itu, pastikan program Anda bekerja dengan benar dan mengatasi masukan yang tidak valid dari pengguna. Setelah selesai membuat program, Anda dapat mengumpulkannya dalam bentuk file Java (berisi kode program) atau file JAR (jika sudah di-compile).