Praktikum Pemrograman Berbasis Objek

Pertemuan 6

Review Time 

1. Polimorfisme
2. Abstract Class
3. Interface
4. UML Interface & Abstract Class
5. Operator instanceof & casting

Materi Pertemuan 6

Gambaran Materi Pembelajaran Hari Ini

Materi Pertemuan 6

01

Type Wrapper

Mengenal wrapper tipe data

02

Enum Class

Mengenal class berisi himpunan konstan

03

Inner Class

Mengenal cara membuat nested class

04

Anonymous Class

Mengenal cara meringkas suatu class

01

Type Wrapper

Type Wrapper

Type Wrapper adalah class yang mengenkapsulasi tipe primitif pada Java. Dibawah adalah type wrapper yang sudah disediakan oleh JDK dalam package **java.lang**.

Primitive	boolean	byte	short	char	int	long	float	double
Wrapper	Boolean	Byte	Short	Character	Integer	Long	Float	Doubleasd

Karena class-class diatas ada pada package **java.lang**, jadi tidak perlu melakukan import manual.

Fungsi Type Wrapper

Type Wrapper dari java.lang bersifat **IMMUTABLE.**

01

Untuk melakukan modifikasi value dari argumen dalam method. Dengan syarat, object harus bersifat Mutable.

02

Class-class dalam package java.util semuanya bekerja dengan object saja. Tidak bisa dengan tipe primitif.

03

Class-class pada Java Collection Framework seperti Vector dan Linked List hanya bisa menyimpan object saja.

04

Konkurensi pada Java seperti multithreading hanya bisa bekerja dengan object.

Mutable Object adalah object yang isi/konten-nya bisa berubah setelah instansiasi. Ketika suatu object Integer ditambahkan dengan angka lain, maka akan dibuat object Integer baru dari hasil operasinya dan **TIDAK MENGUBAH** object yang lama karena wrapper class Integer bersifat **IMMUTABLE**.

Konversi antara Wrapper dan Primitive

Primitive → Wrapper

```
public static void main(String[] args){
    //  Variable primitif
    int pri = 10;

    //  Constructor sudah diharamkan mulai dari Java 9
    Integer obj1 = new Integer(pri);

    //  Pakai factory method
    Integer obj2 = Integer.valueOf(pri);
    Integer obj2 = Integer.valueOf(10);

}
```

Wrapper → Primitive

```
public static void main(String[] args) {
    //  Wrapper Object
    Integer obj1 = Integer.valueOf(10);
    Character obj2 = Character.valueOf('A');

    //  Konversi ke primitif
    int pri1 = obj1.intValue();
    char pri2 = obj2.charValue();
}
```

Autoboxing dan Unboxing

- “Autoboxing” adalah proses boxing yang dilakukan secara otomatis.

Jangan digunakan didalam loop!!! Autoboxing akan memperlambat loop dan mengkonsumsi banyak sumber daya sia-sia.

- “Unboxing” adalah proses ketika value dari type wrapper di-unwrap untuk mendapatkan value dalam tipe primitifnya.

```
public static Integer tambahWrapper(Integer a, Integer b)
    return a + b;
}

public static void main(String[] args) {
    //  Autoboxing
    Integer hasil1 = tambahWrapper(10, 20);
    Integer num1 = 80;

    // Unboxing
    int hasil2 = tambahWrapper(3, 4);
    int num2 = num1;
}
```

Type Wrapper Convenience Method

Convenience Method adalah method yang membuat suatu operasi lebih mudah diakses. Type wrapper dari java.lang memiliki banyak convenience method untuk tipe yang di-wrap nya. Misal untuk Float:

Modifier and Type	Method	Description
byte	byteValue	Returns the value of this Float as a byte after a narrowing primitive conversion.
static int	compare(float f1, float f2)	Compares the two specified float values.
int	compareTo(Float anotherFloat)	Compares two Float objects numerically.
double	doubleValue()	Returns the value of this Float as a double after a widening primitive conversion.
boolean	equals(Object obj)	Compares this object against the specified object.

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Float.html#method.summary

02

Enum Class

Enum Class

Enum adalah sebuah konsep yang tidak eksklusif di Java saja. Hampir seluruh bahasa pemrograman mempunyai konsep enum.

Pada dasarnya, enum adalah sebuah *kumpulan variabel-variabel yang bersifat konstan atau tidak akan berubah*.

Dalam Java, enum class adalah sebuah class yang "spesial". Semua konstanta dalam enum class bersifat **public**, **static** dan **final** (tidak bisa di override).

Enum class bisa saja memiliki method, tapi tidak disarankan. Dan juga enum class **tidak bisa diinstansiasi dan meng-extend class lain**.

```
enum Hari {
    SENIN, SELASA, RABU, KAMIS, JUMAT, SABTU, MINGGU
}

public class Main {
    public static void main(String[] args) {
        Hari prakA = Hari.SELASA;
        Hari prakB = Hari.RABU;
    }
}
```

Switch dan Enum Class

Penggunaan Enum pada Switch

```
public class Main {
    public static void main(String[] args) {
        Hari pbo = Hari.SELASA;

        switch (pbo) {
            case SELASA:
                System.out.println("Prak. Kelas A");
                break;
            case RABU:
                System.out.println("Prak. Kelas B");
                break;
            case default:
                System.out.println("Gaada Praktikum");
        }
    }
}
```

03

Inner Class

Inner Class

Dalam Java, sebuah class dapat memiliki class lagi didalamnya. Class yang berada di dalam class lain disebut inner class atau nested class.

Kenapa menggunakan inner class?

Jika penggunaannya tepat, nested/inner classes membuat kode menjadi lebih mudah untuk dibaca dan dipahami.

```
class OuterClass { // outer class
    int x = 5; // member outer class
    class InnerClass { // inner class
        int y = 10; //member inner class
    }
}

public class Main {
    public static void main(String[] args) {
        OuterClass outer = new OuterClass();

        // instansiasi inner class
        // walaupun outer class punya inner class
        // tidak memiliki objek inner
        OuterClass.InnerClass inner = outer.new InnerClass
        System.out.println(outer.x + inner.y);
    }
}
```

Inner Class Access Modifier

Seperti member lainnya, inner class dapat memiliki access modifier seperti private dan protected.

```
class OuterClass { // outer class
    int x = 5; // member outer class
    private class InnerClass { // private class
        int y = 10; //member inner class
    }
}

public class Main {
    public static void main(String[] args) {
        OuterClass outer = new OuterClass();

        // jika private maka inner class
        // tidak bisa dipanggil
        OuterClass.InnerClass inner = outer.new InnerClass
        System.out.println(outer.x + inner.y);
    }
}
```

Inner class (jika tidak static) dapat mengakses member milik outer class-nya.

```
class OuterClass { // outer class
    int x = 5; // member outer class
    private class InnerClass { // inner class
        public int getOuterX() {
            return x; // mengambil x dari outer
        }
    }
}

public class Main {
    public static void main(String[] args) {
        OuterClass outer = new OuterClass();

        // instansiasi inner class
        OuterClass.InnerClass inner = outer.new InnerClass
        System.out.println(inner.getOuterX()); // 5
    }
}
```

Static Inner Class

Inner class juga dapat didefinisikan secara static. Ini memungkinkan inner class dapat diakses dan diinstansiasi tanpa harus menginstansiasi outer class nya terlebih dahulu.

Sama halnya dengan member static lain. Sebuah static inner class tidak dapat mengakses member non-static dari luar definisinya.

```
class OuterClass { // outer class
    int x = 5; // member outer class
    static class InnerClass { // inner class
        private int y = 10; // member inner class

        public int getY() { // getter y
            return this.y;
        }
    }
}

public class Main {
    public static void main(String[] args) {
        OuterClass.InnerClass inner = new OuterClass.Inner

        System.out.println(inner.getY()); // 10
    }
}
```

04

Anonymous Class

Anonymous Class

Anonymous Class seperti namanya adalah suatu inner class yang tidak memiliki nama.

Apa maksudnya?

Jadi bayangkan ada sebuah class yang ingin diinstansiasi, tapi kita ingin mengubah implementasi dalam suatu methodnya untuk instansiasi ini.

Disini kita tidak usah mendeklarasikan suatu subclass baru dan meng-override method tersebut. Tapi kita bisa menggunakan anonymous class yang meng-extend class tersebut saat instansiasi.

```
class Dog {
    public void speak() {
        System.out.println("guk guk!");
    }
}

public class Main {
    public static void main(String[] args) {
        Dog regularDog = new Dog();

        Dog superDog = new Dog() {
            @Override
            public void speak(){
                System.out.println("rill kh bg?");
            }
        };

        regularDog.speak();
        superDog.speak();
    }
}
```

Instansisasi Anonymous Class

Extend sebuah class yang ada(concrete/abstract).

```
class Base {
    public void printSomething() {
        System.out.println("halo! saya base class");
    }
}

public class Main {
    public static void main(String[] args) {
        Base obj = new Base() {
            @Override
            public void printSomething() {
                System.out.println("halo! saya anon");
            }
        };

        obj.printSomething();
    }
}
```

Implementasi interface.

```
interface Animal {
    public void speak();
}

public class Main {
    public static void main(String[] args) {
        Animal semut = new Animal() {
            @Override
            public void speak() {
                System.out.println("sup bro");
            }
        };

        semut.speak();
    }
}
```

Instansisasi Anonymous Class (lanj.)

Instansiasi pada saat pemberian argumen sebuah method.

```
class SomeClass {
    public void printSomething() {
        System.out.println("something");
    }
}
```

```
public class Main {
    static void execute(SomeClass obj) {
        obj.printSomething();
    }

    public static void main(String[] args) {
        execute(new SomeClass() {
            @Override
            public void printSomething() {
                System.out.println("Overriden print!");
            }
        });
    }
}
```

Instansisasi Anonymous Class (lanj. 2)

Anonymous class juga bisa mengakses variabel yang ada dalam block yang sama.

DENGAN SYARAT variabel tersebut harus final atau “effectively” final.

```
class Base {
    public void printInt(int a) {
        System.out.println("Integer bernilai: " + a);
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        int num = 10;   // "effectively" final
        Base obj = new Base() {
            @Override
            public void printInt(int a) {
                // akses num dalam anon class
                System.out.println(a + num);
            }
        };
        // jika line di bawah diuncomment
        // maka akan error
        // num = 8;
        obj.printInt(5); // 15
    }
}
```

Snippets 👨‍💻

Berikut snippets program materi kali ini

snippets-PBO-05

Silahkan untuk kalian mencoba menjalankan dan mempelajari snippets yang disediakan pada repository tersebut. Selamat belajar!

Exercise!

Semua bisa karena terbiasa

Exercise `Animal.java & Test.java`

Buatlah class `Animal` dengan attribute `String name` beserta setter dan getter dan inner class `Action` dengan method void `move()` dan `speak()` pada file `Animal.java`. Buat juga class `Test` yang berisi method main dari program pada file `Test.java`.

Instansiasikan class Animal ke dalam object `kucing`, lalu instansiasikan inner class nya melalui anonymus class dengan mengoverride method `move()` dan `speak()` dari inner class `Action`. Buat output yang berisi nama, gerakan, dan suara kucing.

OUTPUT Nama, move, dan speak

CONTOH Contoh Output

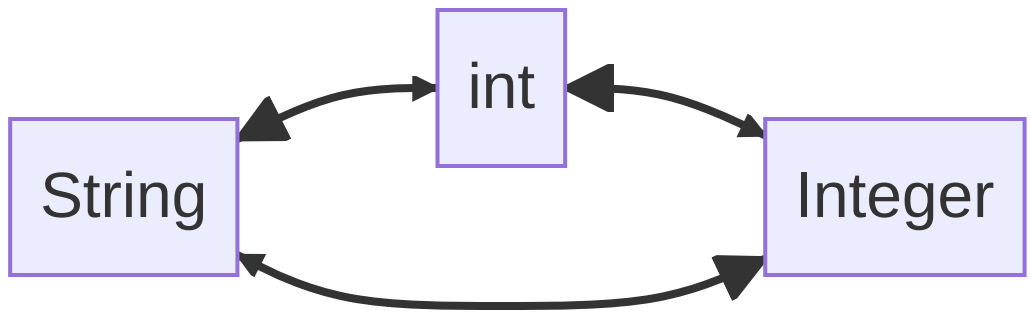
```
Nama : Neko
Move : I am too lazy bruh
Speak : Meow!
```

Assignment!

Seperti biasa, setiap selesai praktikum pasti ada tugas

Assignment 6 Soal 1

Buatlah sebuah program java converter type. Input bisa berupa String, int, dan Integer kemudian bisa diconvert seperti gambar di bawah ini.



*Program dibuat dalam bentuk menu

Assignment 6 Soal 2

Di toko buah terdapat berbagai macam jenis buah, yaitu MANGGA, APEL, ANGGUR, JERUK, dan MELON. Buatlah enum Buah dengan berbagai jenis buah dan harganya. Berikan deskripsi dari buah tersebut pada method getDeskripsi() dan harga pada method getHarga(). Selanjutnya tampilkan data semua buah, harga, dan deskripsinya!

```
public enum Buah {
  MANGGA(/* harga... */), ... ;
  ...

  ...
  public String getDeskripsi() { ... };
  public int getHarga() { ... };
}
```

Assignment 6 Soal 3

Perbaiki program di bawah ini agar bisa di compile dengan tidak mengubah kode pada main method!

```
class Test {
    private final String s;

    static class Inner {
        void testMethod() {
            s = "set from inner";
            System.out.println(s);
        }
    }

    // Don't change the code below!
    // Code below must be executed without error!
    public static void main(String[] args) {
        Test.Inner inner = new Test.Inner();
        inner.testMethod();
    }
}
```

Assignment 6 Soal 4

Implementasi method-method interface di bawah menggunakan Anonymous Class untuk dua (2) bahasa tampilan yang berbeda dan cetak hasilnya. Tulis kode hanya pada main method class Main.java!

```
public interface Greetings {
    public void sayHello(String name);
    public void sayGoodbye(String name);
}
```

Teknis Pengumpulan

Pengerjaan dan pengumpulan tugas akan dilakukan di **Github Classroom**

Kelas A:

Link Tugas Kelas A

Kelas B:

Link Tugas Kelas B

Accept assignment terlebih dahulu lalu link akun Github dengan slot nama yang sesuai di Github Classroom

Teknis Pengumpulan

Format setiap file `.java` didahulukan dengan Nama, NPM, Kelas, Tanggal, dan Deskripsi

Cara menambah comment di java

```
// untuk single line
/* untuk multiple line */
```

Contoh Format

```
/*
  Nama  : Jane Doe
  NPM       : 99
  Kelas     : A
  Tanggal   : 1 September 2021
  Deskripsi : Class jawaban exercise-01 soal-01
*/
```

Deadline Pengumpulan ⌚

Kelas A:

24 Oktober 2022, 23:59 WIB

Kelas B:

25 Oktober 2022, 23:59 WIB

Waktu yang dilihat adalah waktu last commit.
Jika ada yang commit melewati deadline walaupun sudah commit sebelumnya akan dianggap telat

Terima Kasih!

Do you have any questions? Please use respective class discussion channel on Discord.

Key of success adalah kunci kesuksesan 🔥🔥🔥