PRAKTIKUM 15

TYPE WRAPPER

A. TUJUAN PEMBELAJARAN

1. Memahami mengenai type Wrapper dan berbagai macam type Wrapper.
2. Memahami cara perpindahan dari tipe data primitif menjadi tipe Wrapper dan sebaliknya.
3. Memahami proses Autoboxing dan Auto-unboxing.

B. DASAR TEORI

B.1 Pengenalan dan Macam Type Wrapper

Java mengenal 8 buah tipe data primitif dan tidak dapat dibuat objek. Karena hal ini mengalami kesulitan dalam penggunaannya karena beberapa class di library Java hanya dapat berinteraksi dengan objek. Untuk mengatasi masalah ini, Java menyediakan tipe data class untuk tipe data primitif. Class ini membungkus tipe data primitif agar dapat digunakan sebagai objek. Class ini disebut dengan type Wrapper.

Tipe Data Primitif	Class Wrapper	Argument Constructor
Boolean	Boolean	boolean atau String
Byte	Byte	byte atau String
Char	Character	char
Double	Double	double atau String
Float	Float	float, double atau String
Int	Integer	int atau String
Long	Long	long atau String
Short	Short	short atau String

B.2 Number (Class Byte, Short, Integer, Long, Float dan Double)

Number adalah class abstract yang merupakan superclass dari semua class type wrapper yang membungkus tipe data numerik seperti byte, short ,int, long, float dan double. Konstruktor dari class wrapper ini mengizinkan untuk membuat nilai numerik dari objek

String. Namun jika objek String yang dilewatkan sebagai argument konstruktor bukan nilai numerik yang valid, maka melemparkan exception yang bertipe `NumberFormatException`.

Constructor dari Class Byte, Short, Integer, Long, Float dan Double adalah :

```
Byte(byte value)
Byte(String s) throws NumberFormatException
```

Contoh penggunaan

```
Byte num1 = new Byte(100);
Byte num2 = new Byte("100");
```

```
Short(short value)
Short (String s) throws NumberFormatException
```

Contoh penggunaan

```
Short num1 = new Short(100);
Short num2 = new Short("100");
```

```
Integer(int value)
Integer(String s) throws NumberFormatException
```

Contoh penggunaan

```
Integer num1 = new Integer(100);
Integer num2 = new Integer("100");
```

```
Long(long value)
Long(String s) throws NumberFormatException
```

Contoh penggunaan

```
Long num1 = new Long(100);
Long num2 = new Long("100");
```

```
Float(double value)
Float(float value)
Float(String s) throws NumberFormatException
```

Contoh penggunaan

```
Float num1 = new Float(100.4);
Float num2 = new Float(100.4f);
Float num3 = new Float("100.4");
```

```
Double(double value)
Double(String s) throws NumberFormatException
```

Contoh penggunaan

```
Double num1 = new Double(100);
Double num2 = new Double("100");
```

Setiap objek wrapper mempunyai konstanta MAX_VALUE dan MIN_VALUE

MAX_VALUE : nilai numerik terbesar yang dapat ditampung oleh objek dengan tipe data class wrapper ini.

MIN_VALUE : nilai numerik terkecil yang dapat ditampung oleh objek dengan tipe data class wrapper ini.

```
Byte byteObj = new Byte(Byte.MAX_VALUE);
Short  shortObj = new Short(Short.MAX_VALUE);
Integer  intObj = new Integer(Integer.MAX_VALUE);
Long  longObj = new Long(Long.MAX_VALUE);
Float  floatObj = new Float(Float.MAX_VALUE);
Double doubleObj = new Double(Double.MAX_VALUE);
```

Nilai dari maksimum dari objek wrapper adalah sebagai berikut:

```
Byte        :127
Short       :32767
Integer     :2147483647
Long        :9223372036854775807
Float       :3.4028235E38
Double      :1.7976931348623157E308
```

B.3 Method-method pada class wrapper.

- **Method valueOf()**

Method `valueOf()` : mengubah suatu nilai menjadi object dari class tersebut dan melempar NumberFormatException(NFE) jika argument tidak sesuai.

Class Long, Integer, Short dan Byte mempunyai tiga method valueOf()

static Integer	**valueOf**(int i) Returns a Integer instance representing the specified int value.
static Integer	**valueOf**(String s) Returns an Integer object holding the value of the specified String.
static Integer	**valueOf**(String s, int radix) Returns an Integer object holding the value extracted from the specified String when parsed with the radix given by the second argument.

Method 1 menerima argument berupa nilai.

Method 2 menerima argument nilai dalam bentuk String.

Method 3 argument kedua berupa int radix yang menyatakan base dari argument pertama (binary, octal, atau hexadecimal)

Class Boolean, Float dan Double mempunyai dua method valueOf()

static Double	**valueOf**(double d) Returns a Double instance representing the specified double value.
static Double	**valueOf**(String s) Returns a Double object holding the double value represented by the argument string s.

Contoh penggunaan method valueOf()

```
Integer i1 = Integer.valueOf(42);
Integer i2 = Integer.valueOf(“42”);
Integer i3 = Integer.valueOf(“42”,2);
Boolean b1 = Boolean .valueOf(true);
Boolean b2 = Boolean .valueOf(“true”);
Long n1 = Long.valueOf(42000000L);
Long n1 = Long.valueOf(“42000000L”);
```

- **Method xxxValue()**

Method `xxxValue()` digunakan untuk mengubah object dari class wrapper (object ini mempunyai nilai) menjadi nilai numerik

Contoh penggunaan :

```
Integer i2 = new Integer(42);
byte b = i2.byteValue();
short s = i2.shortValue();
double d = i2.doubleValue();
Float f2 = new Float(3.14f);
short s = f2.shortValue();
```

- **Method parseXxx()**

Fungsi `parseXxx()` dan `valueOf()`, argument berupa String dan melempar `NumberFormatException`(NFE) jika argument tidak sesuai. Method `parseXxx()` mengembalikan nilai primitif, sedangkan method `valueOf()` mengembalikan object dari class wrapper

B.4 Character.

Class Character merupakan class wrapper untuk tipe data primitif char. Konstruktor dari class Character adalah :

`Character(char value)`

Method valueOf() digunakan untuk mendapat nilai char dengan tipe wrapper. Character hanya mempunyai satu method valueOf()

`static Character`	**`valueOf`**`(char c)` Returns a `Character` instance representing the specified `char` value.

Method charValue() untuk mendapatkan tipe data primitif char.

`char charValue()`

B.5 Autoboxing / Auto-Unboxing

Pada Java 5, dikenal istilah **`Autoboxing`** dan **`Auto-unboxing`**. **`Autoboxing`** adalah konversi secara otomatis oleh kompiler Java dari tipe data primitif ke tipe data sesuai dengan tipe wrappernya (misalnya, int dan Integer, double dan Double). Sedangkan mengubah object dari class wrapper menjadi nilai primitifnya disebut **`Auto-unboxing.`**

Dengan Autoboxing	**Tanpa Autoboxing**
int i;	**int i;**
Integer j ;	**Integer j ;**
i=1;	**i=1;**
j=2;	**j=new Integer(2);**
i=j;	**i=j.intValue();**
j=i;	**j=new Integer(i);**

Pada kode tanpa Autoboxing, harus secara eksplisit membuat objek Integer j dengan nilai primitif 2, sedangkan dengan autoboxing, maka proses perubahan dari tipe data primitif int secara otomatis akan di konversi menjadi objek Integer, sehingga pada java 1.5 keatas diperbolehkan dari

```
Integer j ;
j = new Integer(2)
```

menjadi

```
Integer j ;
j=2;
```

Contoh dengan Auto-unboxing:

```
Integer y = new Integer(567);
int x = y.intValue();
x++ ;
y = new Integer(x) ;
```

Contoh dengan Autoboxing:

```
Integer y = new Integer(567);
y++;
```

Proses increment biasanya digunakan untuk tipe data primitif. Tapi untuk java 1.5 (dengan Autoboxing), proses increment dapat dilakukan pada objek wrapper. Pada Auto-

unboxing pada waktu hendak melakukan increment, nilai primitifnya harus terlebih dahulu diambil baru kemudian proses increment dapat dilakukan. Hasil increment tersebut kemudian dibuat menjadi objek Integer yang baru.

C. TUGAS PENDAHULUAN

Buatlah review mengenai tipe Wrapper dan macam-macamnya, dan penggunaan konstruktor dari masing-masing tipe Wrapper.

D. PERCOBAAN

Percobaan 1 : Memahami cara membuat objek Integer. Jika terjadi error atau exception, jelaskan penyebabnya !

```
public class TestInteger {
    public static void main(String[] args) {
        Integer i1 = new Integer(42);
        Integer i2 = new Integer(2147483647);
        Integer i3 = new Integer("42");
        Integer i4 = new Integer("42.u");
    }
}
```

```
public class TestInteger {
    public static void main(String[] args) {
        Integer i1 = new Integer(42);
        Integer i2 = new Integer(2147483650);
        Integer i3 = new Integer("42");
        Integer i4 = new Integer("42.u");
    }
}
```

Percobaan 2 : Mengetahui nilai maksimum dan minimum untuk tipe Wrapper.

```
public class MaxMin {
    public static void main(String[] args) {
        System.out.println("Nilai Maximum");
        Byte byteObj = new Byte(Byte.MAX_VALUE);
        System.out.println(byteObj);
        Short  shortObj = new Short(Short.MAX_VALUE);
        System.out.println(shortObj);
        Integer  intObj = new Integer(Integer.MAX_VALUE);
        System.out.println(intObj);
        Long  longObj = new Long(Long.MAX_VALUE);
        System.out.println(longObj);
        Float  floatObj = new Float(Float.MAX_VALUE);
        System.out.println(floatObj);
        Double doubleObj = new Double(Double.MAX_VALUE);
```

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

        System.out.println("Nilai Minimum");
        Byte byteObj2 = new Byte(Byte.MIN_VALUE);
        System.out.println(byteObj2);
        Short  shortObj2 = new Short(Short.MIN_VALUE);
        System.out.println(shortObj2);
        Integer  intObj2 = new Integer(Integer.MIN_VALUE);
        System.out.println(intObj2);
        Long  longObj2 = new Long(Long.MIN_VALUE);
        System.out.println(longObj2);
        Float  floatObj2 = new Float(Float.MIN_VALUE);
        System.out.println(floatObj2);
        Double doubleObj2 = new Double(Double.MIN_VALUE);
        System.out.println(doubleObj2);
    }
}
```

Percobaan 3 : Menampilkan bilangan integer menjadi biner, octal dan heksadesimal.

```
public class TestInteger2 {
    public static void main(String[] args) {
        Integer i1 = new Integer(345);
        System.out.println("nilai int = " + i1);
        System.out.println("bil        binary        =        "        +
Integer.toBinaryString(i1));
        System.out.println("bil hexa = " + Integer.toHexString(i1));
        System.out.println("bil oktal = " + Integer.toOctalString(i1));

        Integer i2 = new Integer(675);
        System.out.println("nilai int = " + i2);
        System.out.println("bil binary = "+Integer.toString(i2, 2));
        System.out.println("bil hexa = " +Integer.toString(i2, 8));
        System.out.println("bil oktal = " + Integer.toString(i2, 16));
    }
}
```

Percobaan 4 : Memahami cara konversi antar tipe Wrapper, misal dari objek Integer diassignkan ke objek Long dan sebaliknya.

```
public class Konversi {
    public static void main(String[] args) {
        Integer i = 7 ;
        Long l = 345L ;

        i = l.intValue() ;
        System.out.println("Nilai pada Integer = " + i);

        l = i.longValue() ;
        System.out.println("Nilai pada Long = " + l) ;
    }
}
```

Percobaan 5 : Jelaskan kegunaan dari method-method di bawah ini!

```
public class WrapperMethod {
    public static void main(String[] args) {
        System.out.println(Integer.rotateLeft(3, 1));
        System.out.println(Integer.rotateLeft(5, 2));
        System.out.println(Integer.rotateRight(20, 1));
        System.out.println(Integer.rotateRight(32, 1));
        System.out.println(Integer.reverse(20));
        System.out.println(Integer.highestOneBit(20));
        System.out.println(Integer.lowestOneBit(20));
        System.out.println(Integer.bitCount(20));
        System.out.println(Integer.numberOfLeadingZeros(32));
        System.out.println(Integer.numberOfTrailingZeros(32));
    }
}
```

Percobaan 6 : Memahami mengenai Autoboxing dan Auto-unboxing.

```
public class TestUnboxing {
    public static void main(String[] args) {
        int i;
        Integer j;
        i=1;
        j=new Integer(2);
        i=j.intValue();
        j=new Integer(i);
    }
}
```

```
public class TestAutoboxing {
    public static void main(String[] args) {
        int i;
        Integer j;
        i=1;
        j=2;
        i=j;
        j=i;
    }
}
```

Percobaan 7 : Memahami mengenai Autoboxing dan Auto-unboxing(2)

```
public class TestUnboxing2 {
    public static void main(String[] args) {
        Integer y = new Integer(567);
        int x = y.intValue();
        System.out.println("x = " + x);
        x++ ;
        y = new Integer(x) ;
        System.out.println("y = " + y);
    }
```

```
}
```

```
public class TestAutoboxing2 {
    public static void main(String[] args) {
        Integer y = new Integer(567);
        System.out.println(y);
        y++;
        System.out.println(y);
    }
}
```

E. LATIHAN

1. Terdapat program seperti di bawah ini !

```
class Hexy {
  public static void main(String[] args) {
    Integer i = 42;
    String s = (i<40)?"life":(i>50)?"universe":"everything";
    System.out.println(s);
  }
}
```

What is the result?

A. null
B. life
C. universe
D. everything
E. Compilation fails.
F. An exception is thrown at runtime.

2. Given:

```
1. class Example {
2.   public static void main(String[] args) {
3.     Short s = 15;
4.     Boolean b;
5.     // insert code here
6.   }
7. }
```

Which, inserted independently at line 5, will compile? (Choose all that apply.)

```
A. b = (Number instanceof s);
B. b = (s instanceof Short);
C. b = s.instanceof(Short);
D. b = (s instanceof Number);
```

```
E. b = s.instanceof(Object);
F. b = (s instanceof String);
```

3. Given:

```
class  Fork {
  public static void main(String[] args)    {
    if(args.length == 1 | args[1] .equals("test")) {
      System.out.println ("test  case");
    } else {
      System.out.println("production " + args[0]) ;
    }
  }
}
```

And the command-line invocation:

```
java Fork live2
```

What is the result?

A. test case
B. production
C. test case live2
D. Compilation fails.
E. An exception is thrown at runtime.

4. Given:

```
class Foozit {
  public static void main(String[] args) {
    Integer x = 0;
    Integer y = 0;
    for(Short z = 0; z < 5; z++)
      if((++x > 2) || (++y > 2))
        X++ ;
    System.out.println (x + " " + y);
  }
}
```

What is the result?

```
A. 5 1
B. 5 2
C. 5 3
D. 8 1
E. 8 2
F. 8 3
G. 10 2
H. 10 3
```

5. Given:

```
class Titanic {
  public static void main(String[] args) {
    Boolean bl = true;
    boolean b2 = false;
    boolean b3 = true;
    if((bl & b2) | (b2 & b3) & b3)
      System.out.print("alpha ");
    if((bl = false) | (b1 & b3) | (bl | b2))
      System.out.print("beta "};
  }
}
```

What is the result?

A. beta
B. alpha
C. alpha beta
D. Compilation fails.
E. No output is produced.
F. An exception is thrown at runtime

```
6. class Feline {
  public static void main(String[] args) {
    Long x = 42L;
    Long y = 44L;
    System.out.print (" " + 7 + 2 + " ") ;
    System.out.print(foo () + x + 5 + " ");
    System.out.println(x + y + foo());
  }
  static String foo() { return "foo"; }
}
```

What is the result?

A. 9 foo47 86foo
B. 9 foo47 4244foo
C. 9 foo425 86foo
D. 9 foo425 4244foo
E. 72 foo47 86foo
F. 72 foo47 4244foo
G. 72 foo425 86foo
H. 72 foo425 4244foo
I. Compilation fails.

7. Buatlah sebuah aplikasi untuk mengubah sebuah bilangan desimal menjadi bilangan biner, octal dan heksa berdasarkan inputan user.

```
Input
Masukkan bilangan : 345

Output
biner : 101011001
oktal : 531
heksa : 159
```

8. Buatlah sebuah aplikasi menerima inputan user berupa bilangan dan basis dari bilangan tersebut, selanjutnya mengubah ke bilangan dengan basis yang lain (basis 2,8,10,16)

```
Input
Masukkan bilangan : 531
Basis : 8

Output
biner : 101011001
desimal : 345
heksa : 159
```

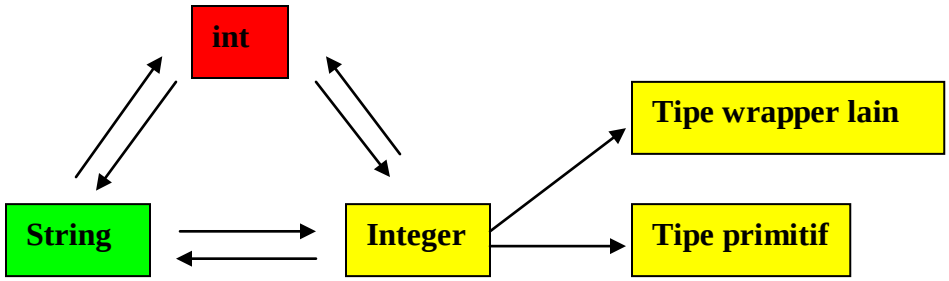
9. a. Bagaimana program ini jika dijalankan? Jelaskan!
   b. Bagaimana cara mengubah objek Long menjadi objek Integer dan sebaliknya ?

```
public class Coba {
    public static void main(String args[]){
         Float f = 2.3f ;
         Double d = 34.7 ;
         i = j ;
    }
}
```

10. Ubahlah object Double menjadi tipe data primitif byte, short, int, long, float.

F. TUGAS

1. Buatlah suatu aplikasi untuk mengubah :



2. Buatlah sebuah program untuk menerima inputan beberapa bilangan(Subclass Number) dari command line dan menambahkannya.

```
java Adder 1 3 2 10
Output : 16
```

G. LAPORAN RESMI

Kerjakan hasil percobaan(D) dan latihan(E) di atas dan tambahkan analisa.