

Java Data Types & Variable

Silver - Chapter 1 - Topic 3

Selamat datang di **Chapter 1 Topic 3**
online course **Back End Java** dari
Binar Academy!

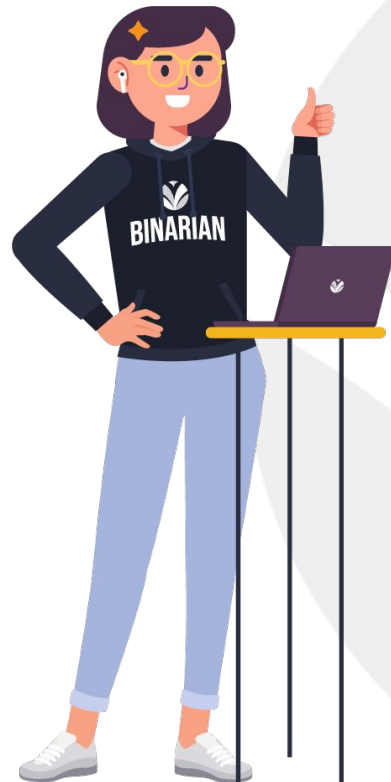


Ketemu lagi ya, Binarian! 🎉

Gimana kabar kamu? Setelah mempelajari Hello World pada topik sebelumnya, sekarang kita move on ke pembelajaran seputar Data dan Variables.

Mulai dari Konsep Variable, Tipe Data Primitive, Cara Membuat methods sampai ke Debugging error dengan IDE, kita akan kenalan sama semuanya.

Oh iya, abis ini ada peta materi yang bisa kamu intip supaya tahu perjalanan materi yang dilalui. Go go go!



Detailnya, kita bakal bahas hal-hal berikut ini:

- Konsep Variable
- Tipe Primitive Data
- Bagaimana cara membuat methods
- Melakukan basic input-output data
- Perbedaan runtime error & compile time error
- Melakukan Debugging error dengan IDE



Pertama bin utama, kita akan mulai dengan materi **Variable**.

Bukan, bukan. Ini nggak ada kaitannya sama matematika kok. Ini lebih seru dari itu 😏



Variable adalah tempat buat menyimpan sebuah data~

Ibarat makan bakso, nih. Mangkok bisa kita sebut sebagai variable, sedangkan baksonya adalah data.

Kebayang, kan?

Ohiya pada chapter sebelumnya, deklarasi variable buat tipe data String dan int sebenarnya udah pernah kita lakuin, kok.

Jadi, kamu bisa cek lagi untuk menambah pemahaman kamu!



VARIABEL



DATA

4 variabel yang biasa dikenal di Java

Nggak mau kalah sama matematika, Java juga punya variable. Tapi bukan variabel x dan y ya!

Variable tersebut yaitu, **Instance Variable** dan **Class Variable**



Diantaranya:

1. **Instance Variable**, yaitu variable yang dideklarasikan di dalam suatu class, di luar method content tanpa modifier static.
2. **Class Variable**, yaitu variable yang dideklarasikan di dalam suatu class, di luar method content tanpa modifier static.

Bedanya dengan instance variable adalah, pada class variable cuma ada satu salinan variable yang dibagikan dengan semua class variable.



Lanjutan nih bun~

3. **Local Variable**, yaitu variable yang dideklarasikan di dalam method content.
4. **Parameter**, yaitu parameter suatu fungsi juga dianggap suatu variable.





Sebelum kupas tuntas contoh dari keempat variabel.

Perhatikan kuis dadakan berikut!

Coba tebak, dari keempat jenis variable tadi, variable mana yang udah pernah ada di topic 2?

```
public class HelloWorld {  
    private String anyField = "Halo";  
  
    private static void printSomething() {  
        System.out.println(anyField);  
    }  
}
```

Variable anyField bisa dipakai dan dimodifikasi oleh seluruh method yang ada di class HelloWorld.

```
public class HelloWorld {  
    private static int sum () {  
        int angka1 = 10;  
        int angka2 = 20;  
        return angka1 + angka2;  
    }  
}
```

Variable **angka1** dan **angka2** cuma bisa dipakai dan dimodifikasi oleh **method sum**.

```
public class HelloWorld {  
  
    public static void main(String[] args) {  
        printSomething("Balonku");  
        sum(10, 10);  
    }  
  
    private static void printSomething(String anyString){  
        System.out.println(anyString);  
    }  
  
    private static int sum (int anyInteger1, int anyInteger2){  
        return anyInteger1 + anyInteger2;  
    }  
}
```

Variable anyString cuma bisa diakses dan dimodifikasi oleh **method content printSomething**.

Begitu juga dengan **anyInteger1** dan **anyInteger2** hanya bisa diakses dan dimodifikasi **method content sum**.

```
public class HelloWorld {  
    public static void main(String[] args) {  
        printSomething(5);  
    }  
  
    private static void printSomething(int anyInteger) {  
        anyInteger = 10;  
        System.out.println(anyInteger);  
    }  
}
```

Value dari suatu variable bisa diubah dengan cara melakukan reassign terhadap variable tersebut.

Parameter variable ini diassign oleh argument dengan value 5, terus di dalam method printSomething, variable tersebut dilakukan reassign dengan value 10.

Jadi, anyInteger saat ini punya value 10

```
public class Main {  
    public static void main(String[] args) {  
        int i = 2;  
        doThis(i);  
        System.out.println(i);  
    }  
  
    public static void doThis(int i) {  
        i = 3;  
    }  
}
```

Coba tebak, apa yang bakal keluar di console?

```
public class Main {  
    public static void main(String[] args) {  
        int i = 2;  
        i = doThis(i);  
    }  
  
    public static int doThis(int i) {  
        return 3;  
    }  
}
```

Satu lagi nih, kira-kira apa yang bakal keluar di console?

Terus, apa kesimpulannya?

Primitive Data Types~

Eh, eh, ini juga bukan pelajaran IPS ya.

Biar nggak ada kesalahpahaman, mari kita intip penjelasannya!

Cekidot~



Sebenarnya data types itu apa, sih?

Data types atau tipe data adalah sebuah pengklasifikasian data berdasarkan jenis data tersebut.

Biasanya, data types ini tersimpan dalam variable. Salah satu jenis data types pada Java adalah primitives data types.



“Duh, istilah apa lagi tuh?”

Jadi, **primitive data types** adalah tipe data paling dasar pada Java.

Tipe ini udah didefinisikan oleh Java dan diberi nama dengan kata kunci. Misalnya, int, long, char, dsb.

Penasaran sama jenis primitive data types? Yuk kita lanjut~



Berikut 8 primitive data types pada Java!

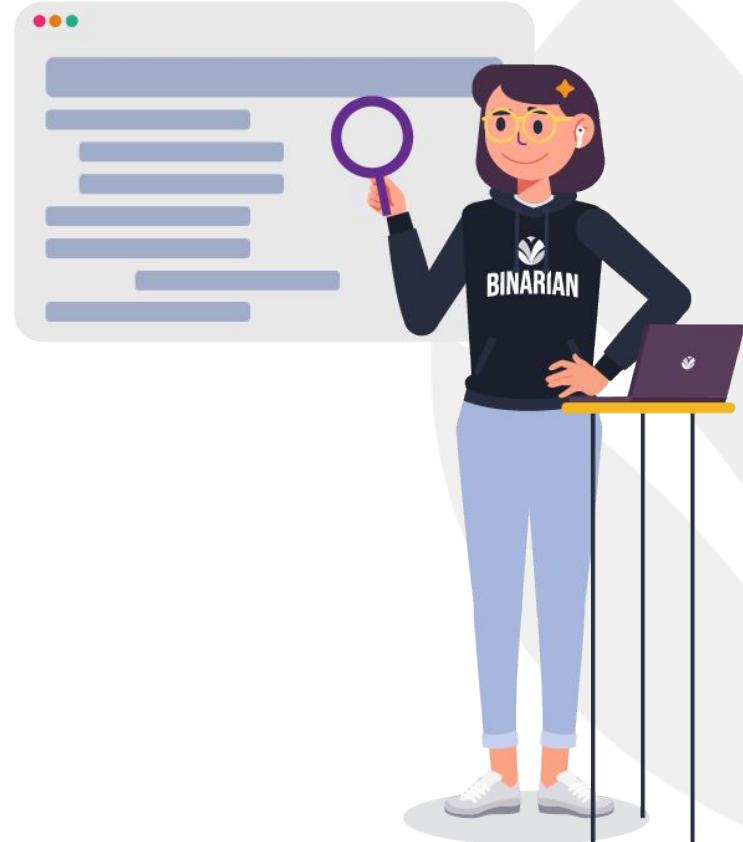
| Tip | data | Keterangan Ukuran (bit) |
|---------|---------------------------------|-------------------------|
| byte | Byte-size integer | 8 |
| short | Short integer 16 | 16 |
| int | Integer | 32 |
| long | Long Integer | 64 |
| char | Single Character | 16 |
| float | Single-Precision Floating Point | 32 |
| double | Double-Precision Floating Point | 64 |
| boolean | True Or False | 1 |

Bukan cuma cerita rakyat yang punya moral value, primitive data juga punya istilah yang disebut default value, lho!

Value kan artinya nilai.

Kalau di Java, default nilai atau value dari primitive data type yang berupa angka adalah 0, sedangkan default value dari Boolean adalah false.

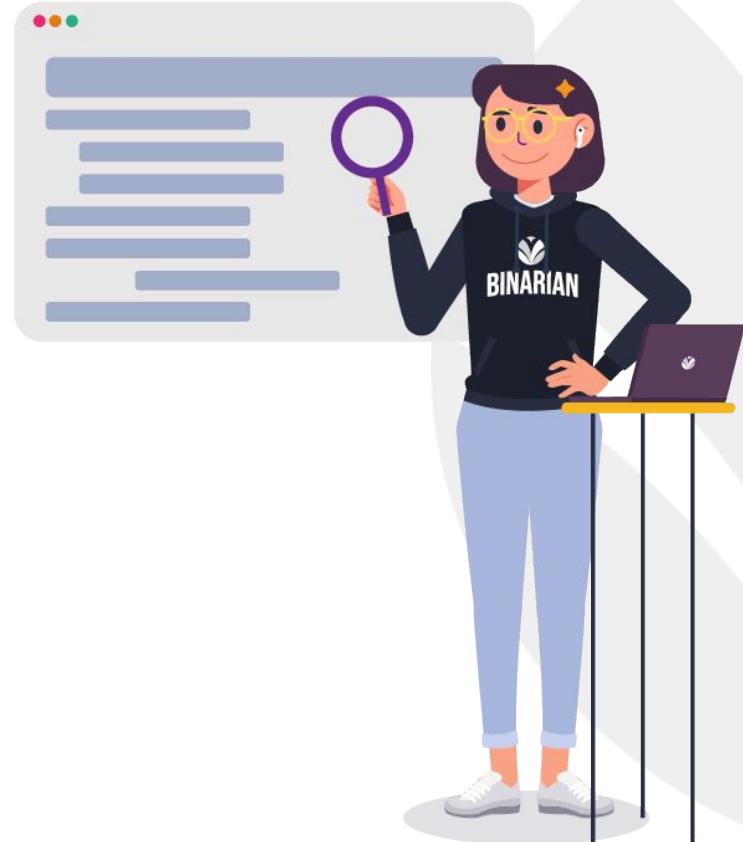
Dengan kata lain, **primitive data type** nggak akan pernah memiliki nilai null.



“Terus primitive data type dipakainya kapan?”

Ketika sebuah variable yang dideklarasikan nggak di assigned, maka default value yang bakal jadi valuenya.

Mirip-mirip deh kayak istilah, cinta ditolak dukun bertindak. Tapi, nggak seseram itu ya!



Tadi kan udah di spoiler duluan tentang short, int, dan long. Sekarang kita bahas satu-satu mengenai primitive data types ya!

4 tipe primitive data yang pertama, yaitu **byte**, **short**, **int** dan **long**.

Perbedaannya adalah di rentang nilainya dan size datanya. **Semakin besar size datanya, maka rentang nilainya semakin panjang.**



Detail perbedaan primitive data types!

| Tipe Data | Panjang | Rentang Nilai | Contoh Nilai |
|-----------|---------|--|---------------------------|
| byte | 8 bit | -2^7 sampai $2^7 - 1$ (-128 sampai 127) (256 kemungkinan nilai) | 5 -126 |
| short | 16 bit | -2^{15} sampai $2^{15} - 1$ (-32.768 sampai 32.767) (65.535 kemungkinan nilai) | 9 -23659 |
| int | 32 bit | -2^{31} sampai $2^{31} - 1$ (-2.147.483.648 sampai 2.147.483.647) (4.294.967.296 kemungkinan nilai) | 2067456397 -1456398567 |
| long | 64 bit | -2^{63} sampai $2^{63} - 1$ (-9.223.372.036.854.775.808 sampai 9.223.372.036.854.775.807) (18.446.744.073.709.551.616 kemungkinan nilai) | 3L -2147483648L 67L |

Selain primitive data, ada juga yang namanya tipe data floating point dan double

Tipe data **floating point** dan **double** adalah tipe data buat variabel yang nilainya adalah bilangan real (bisa punya pecahan desimal).

Kalau di matematika, bilangannya kayak angka -1, -0.5, 0, 0.5, dan 1.

| Tipe Data | Panjang | Contoh Penulisan Nilai yang Diperbolehkan |
|-----------|---------|---|
| float | 32 bit | 78F -34736.86F 6.4E4F (sama dengan $6,4 \times 10^4$) |
| double | 64 bit | -2356 3.5E7 67564788965.567 |

Masih bingung? ini perbedaan diantara keduanya, Gengs!

Perbedaannya adalah pada jenis bilangan.

Byte, short, int dan long menggunakan **angka bulat**.

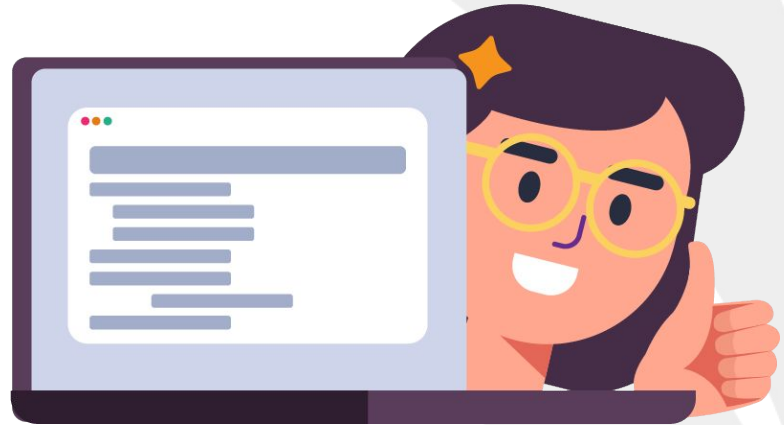
Sedangkan floating dan double menggunakan **angka desimal**.



Primitive data selanjutnya adalah Char~

Bukan mobil ya, gengs. Kalo mobil, bahasa inggrisnya car, kalau yang ini Char, pake h.

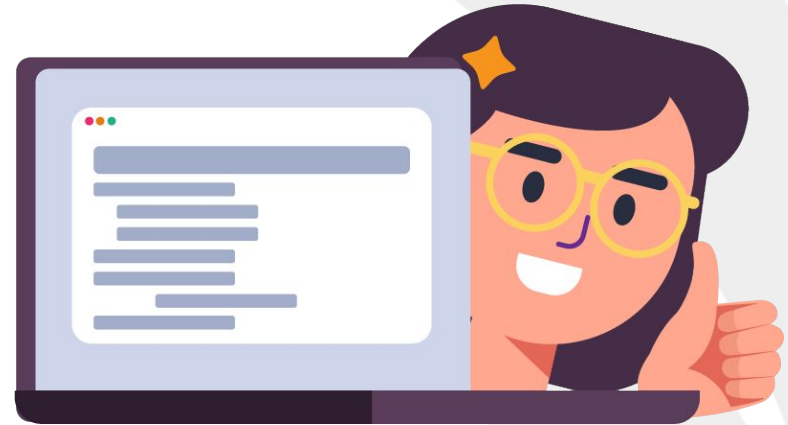
Char atau tipe data textual merupakan tipe data buat variabel yang nilai-nilainya adalah karakter tunggal.



Biasanya, Char berupa alphabet atau dalam bentuk **ascii**. Tipe data textual adalah char yang memiliki panjang 16 bit. Nilai variabel char ditulis dengan diberi tanda kutip tunggal '...'.
✦

Berikut contoh penggunaan tipe data char:

```
public char alphabet = 'A';  
  
public char ascii = '\\111'; // jika dicetak,  
akan menghasilkan  
  
// huruf 'I';
```





Ada nyonya, jatuh terluka. Data types selanjutnya, adalah logika~

Primitive data types logika biasa disebut juga sebagai Boolean.

Boolean ya, bukan roti bolen yang dalamnya ada pisang 🤪

```
public boolean status = true;

    public boolean check = 10 < 5 ; //
nilai check menjadi
    // false
```



Tipe data logika adalah tipe data yang cuma punya 2 kemungkinan nilai, yaitu true atau false.

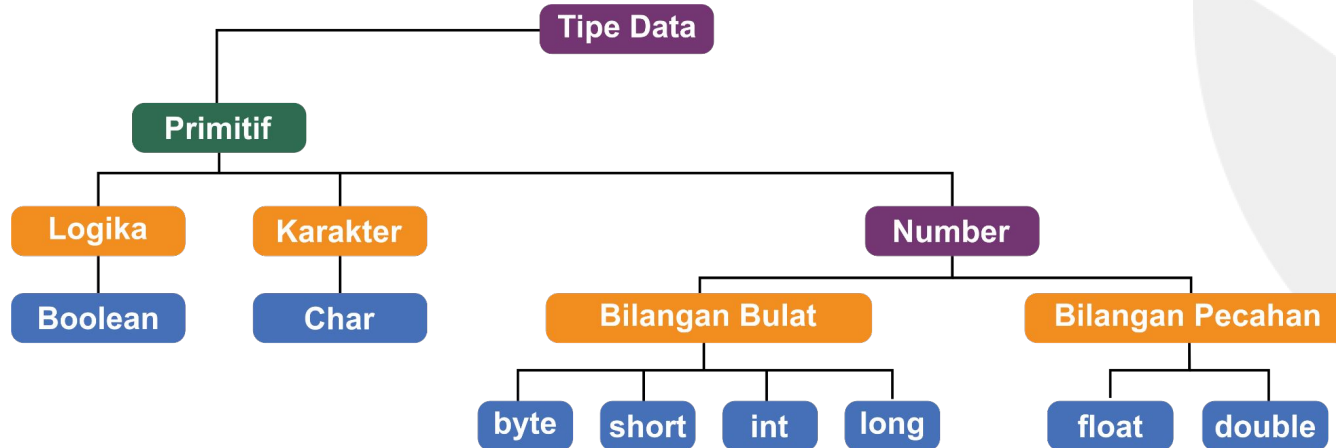
Selain itu, **Boolean adalah satu-satunya tipe data logika yang ada di Java.** Berikut contoh penggunaan tipe data Boolean pada Java:

```
public boolean status = true;

        public boolean check = 10 < 5 ; //
nilai check menjadi
        // false
```

Bukan sulap bukan sihir, dan biar nggak pusing, berikut ada kategorisasi dari 8 tipe data primitive!

Jadi, 8 tipe data primitive yang udah dibahas sebelumnya, ternyata bisa dibagi jadi **3 kelompok**, yaitu: **number** (bilangan bulat & desimal), **Karakter**, dan **Logika**.



Okey, biar nggak makin penasaran.

Langsung aja kita cus masuk ke materi
**Method with Various Return and
Parameter Type.**

Let's go~



Sebenarnya, method bisa dipakai di berbagai tipe data pada return ataupun parameternya, lho!

“Maksudnya gimana?”

Jadi, pada method yang multi parameter, method bisa dipakai di lebih dari satu jenis data types.

Contohnya kaya di samping ini, nih.

```
private static void printSomething(int anyInteger, char anyChar){  
    System.out.println(anyInteger + anyChar);  
}
```

Di sana gunung, di sini gunung.

Di tengah-tengahnya ada siput.

“Cakep!”

Jangan bingung, jangan linglung.

Sekarang kita masuk ke materi **Basic Input Output!**

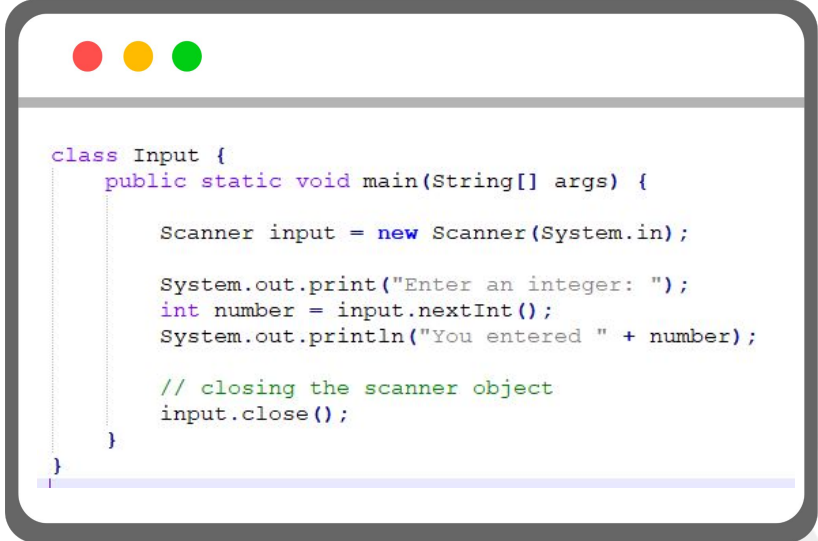


Yang sederhana tapi banyak caranya~

Input adalah proses yang dilakukan buat memasukan data dari screen.

Buat melakukan input pada Java, bisa menggunakan kode **java.util.scanner**.

Contoh penggunaannya, kayak gambar di samping ya~




```
class Input {  
    public static void main(String[] args) {  
  
        Scanner input = new Scanner(System.in);  
  
        System.out.print("Enter an integer: ");  
        int number = input.nextInt();  
        System.out.println("You entered " + number);  
  
        // closing the scanner object  
        input.close();  
    }  
}
```

Kalau ada Input, pasti ada Output dong~

Ibarat mau isi bensin di pertamimin, nggak mungkin dong jalan masuk dan keluarnya disamakan?

Itulah fungsinya tanda arah masuk dan keluar. Karena kalo disamakan, yang ada malah jadi acak-acakan deh tuh kendaraan yang ada.

Kalau di Java, nih, **output dilakukan untuk menampilkan sesuatu di screen**. Cara yang harus kita lakukan yaitu pakai kode perintah di samping, ya.



```
System.out.println();  
  
System.out.print();  
  
System.out.printf();
```

Jalan rahasia buat pakai Input dan Output!

Sebetulnya, ada cara lain yang bisa dipakai buat melakukan Input dan Output, lho. Yaitu pake **BufferedWriter**.

Pernah dengar?

Kalau belum, tenang aja. Nanti bakal kita bahas bareng-bareng di topic Working With Media and PDF di topic 5 ya!



Sama kayak manusia yang suka mumet kalo lagi pusing. Java juga ngerasain hal yang sama.

Bahasa kerennya sih Error.

Nah, di sini kita bakal cari tau **Error Types** yang dirasakan sama si Java.



Ada dua error types, yaitu Runtime Error dan Compile Time Error. Apa sih artinya?

- **Runtime Error**, yaitu error yang terjadi setelah kode dieksekusi.

Kayak ada istilah yang bilang “nggak makan lemes, abis makan malah mager”



- **Compile Time Error**, yaitu error yang terjadi saat kode dicompile, jadi aplikasi nggak sempat jalan sama sekali.

Ibarat udah bikin wacana jalan-jalan, eh jadi gagal karena ada kenaikan status PPKM.



Jadi, apa sih bedanya Runtime Errors dan Compile Time Errors?

| Runtime Errors | Compile Time Errors |
|--|--|
| Compiler nggak akan menemukan error saat aplikasi di compile, tetapi error baru akan terdeteksi pada saat mengeksekusi kode yang salah | Compiler dengan mudah mendeteksi compile-time error selama development dari code tersebut |
| Penyebab runtime error lebih kompleks | Error berkaitan dengan semantics dan syntax |
| Setelah menemukan kesalahan runtime, kode bisa diperbaiki. Namun untuk mengetes fungsinya, aplikasi harus dijalankan terlebih dahulu dan mengulang kembali pada bagian yang terjadi runtime error sebelumnya | Karena aplikasi tidak berjalan saat di kompilasi, dengan melakukan kompilasi, kesalahan dapat langsung dideteksi pada baris ke berapa pada console |

Biar makin paham, kita masuk ke contohnya ya!

```
class MisspelledVar {  
    public static void main(String args[])  
    {  
        int a = 40, b = 60;  
  
        // Declared variable Sum with Capital S  
        int Sum = a + b;  
  
        // Trying to call variable Sum  
        // with a small s ie. sum  
        System.out.println(  
            "Sum of variables is "  
            + sum);  
    }  
}
```

```
prog.java:14: error: cannot find symbol  
            + sum);  
              ^  
symbol:   variable sum  
location: class MisspelledVar  
1 error
```

Nggak cukup dua? Oke, ini masih contoh dari Compile Time Error~

```
class IncorrectLoop {  
    public static void main(String args[])  
    {  
  
        System.out.println("Multiplication Table of 7");  
        int a = 7, ans;  
        int i;  
  
        // Should have been  
        // for(i=1; i<=10; i++)  
        for (i = 1, i <= 10; i++) {  
            ans = a * i;  
            System.out.println(ans + "\n");  
        }  
    }  
}
```

```
prog.java:12: error: not a statement  
        for (i = 1, i <= 10; i++) {  
                ^  
prog.java:12: error: ';' expected  
        for (i = 1, i <= 10; i++) {  
                                ^  
2 errors
```

Ini adalah contoh dari Compile time error. Perhatikan ya, Gengs!

```
class PrintingSentence {  
    public static void main(String args[])  
    {  
        String s = "Java Back End Class";  
  
        // Missing ';' at the end  
        System.out.println("Welcome to " + s)  
    }  
}
```

```
prog.java:8: error: ';' expected  
        System.out.println("Welcome to " + s)  
                                   ^  
1 error
```

Nah, kalau yang ini contoh dari Runtime error dengan pembagian dengan angka 0~

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at DivByZero.main(File.java:14)
```

```
class DivByZero {
    public static void main(String args[])
    {
        int var1 = 15;
        int var2 = 5;
        int var3 = 0;
        int ans1 = var1 / var2;

        // This statement causes a runtime error,
        // as 15 is getting divided by 0 here
        int ans2 = var1 / var3;

        System.out.println(
            "Division of var1"
            + " by var2 is: "
            + ans1);
        System.out.println(
            "Division of var1"
            + " by var3 is: "
            + ans2);
    }
}
```

Masih ada juga nih contoh dari Runtime Error yang lainnya!

1. **Parsing**, yaitu **kesalahan saat berusaha melakukan parsing atau penguraian**.

Contohnya kamu mau mengambil nama file dari pattern nama_file.jpg.

Tapi, pas input data, file yang masuk malah berupa nama_file.mpeg.

Kalau dilihat dari ekstensi filenya, bukannya file berupa gambar yang kamu ambil, tapi malah file video.



2. **Out of Bounds**, yaitu **kesalahan saat memanggil array** dengan index melebihi dari size array
3. **Null Pointer**, yaitu **saat sebuah object null memanggil methodnya**



Kalau ada yang error, pasti harus diperbaiki kan?

Oleh karena itu, mari kita sambut materi penutup kita, yaitu **Debugging Error With IDE~**

Yeayyy!

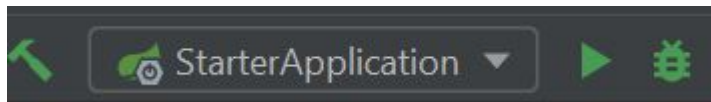


Karena segala kesalahan harus segera diperbaiki~

Debugging merupakan proses pendeteksian dan penghapusan dari bug.

Pasti kamu sering denger 'bug' kan?

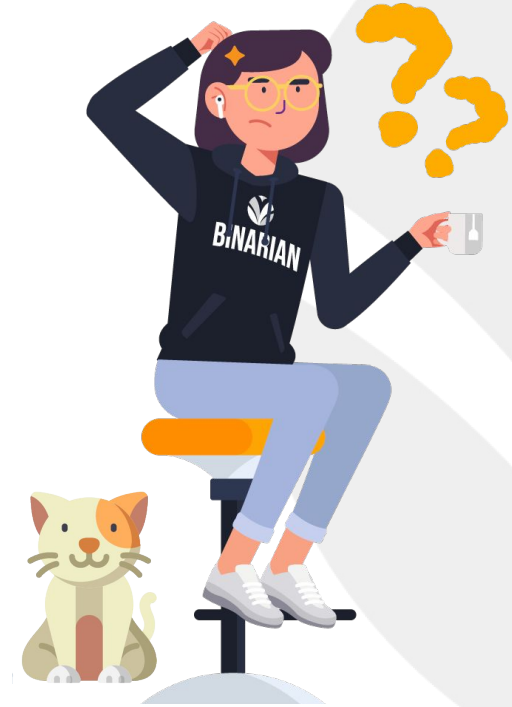
Yup, bug adalah temuan yang sifatnya nggak sesuai dengan ekspektasi. Biasanya, debug dilambangkan dengan gambar serangga.



“Cara tahu kalau kita lagi mengalami bug, gimana sih?”

Tenang, tenang~

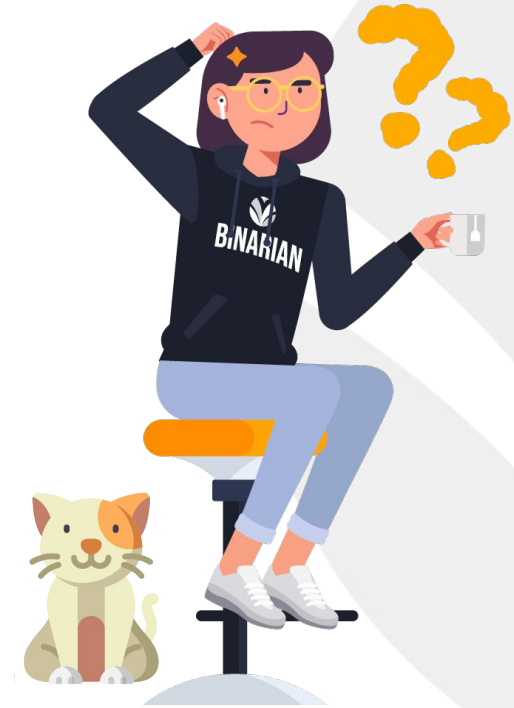
Buat mencari tahu bugs yang ada, biasanya developer pakai **console**, **log**, dan **history** di database atau bisa juga pakai **fitur debugging** pada IDE.



Kalau pake IDE, segala runtime error bisa cepat dideteksi dan diselesaikan.

Kalau nggak, seringkali developer cuma bisa menebak alur eksekusi kode programnya untuk mencari tahu di proses mana yang menyebabkan error/bug.

Jadi makin pusing, kan?



**Kalau udah banyak yang pakai,
pasti jelas bermanfaat, kan?**

Dengan fitur debugging pada IDE, developer bisa membuat breakpoint di line yang berisi statement.

Hal ini yang membantu developer menemukan root cause atau akar masalah.



Saat statement di mana kita bikin breakpoint bakal dieksekusi, kita bisa mengecek nih value dari variable yang ada.

Breakpoint berfungsi buat penanda supaya eksekusi kode program pas proses debugging harus dihentikan sebentar di titik tersebut.

Canggih banget, kan?



Berikut adalah contoh dari pembuatan breakpoint~



```
39 List<InetSocketAddress> addresses = new ArrayList();  
40 Iterator var3 = urls.iterator();
```

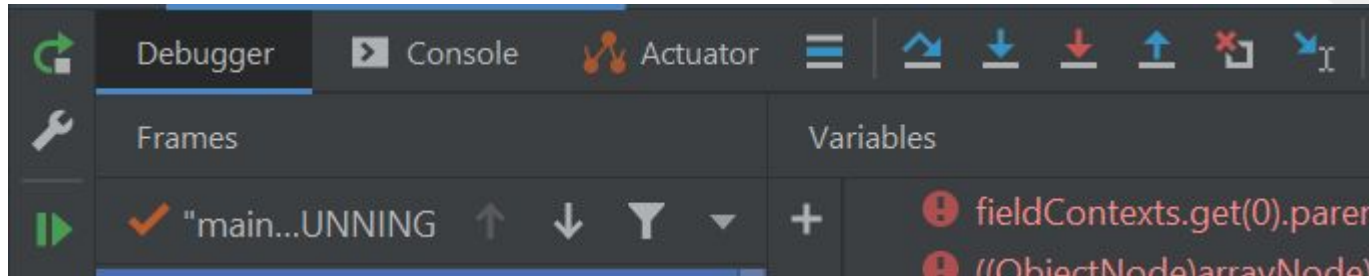
Kamu lihat titik bulat merah di atas?
Nah, titik merah tersebut adalah breakpoint-nya.

Cara buatnya, kamu hanya tinggal men-double click pada baris source code. Gampang banget kan, pastinya~

Oh, iya!

Posisi breakpoint sebaiknya udah ditentukan dari runtime error yang sebelumnya, ya, biar kita bisa mencari root causenya.

Buat melanjutkan eksekusi, kita bisa coba beberapa opsi, lho!



Tadi kamu udah belajar 8 tipe data primitive yang terdiri dari number (bilangan bulat & desimal), karakter, dan logika.

Kalo dipikir-pikir lagi, kenapa sih di bahasa pemrograman Java itu setiap variable harus ditulis bakal berisi tipe data apa?



Nah, selesai sudah pembahasan kita di topic 3 ini.

Selanjutnya, kita bakal bahas tentang **Operator, Conditional & Looping**

Penasaran kayak gimana? Cus langsung ke topik selanjutnya~

