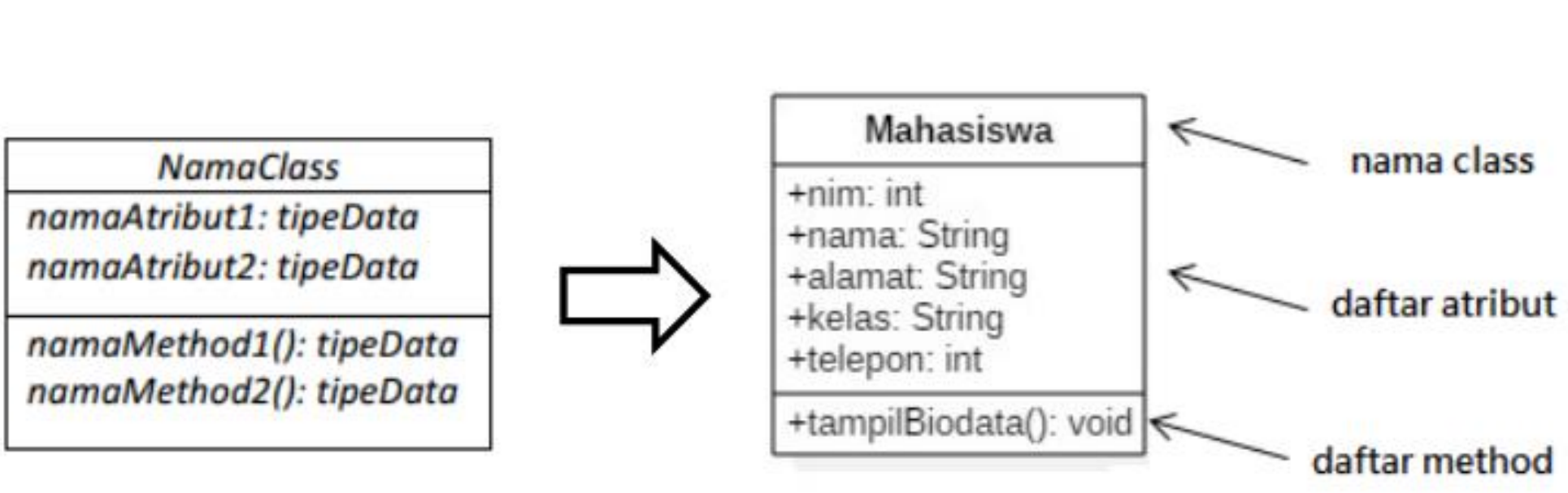
Class dan Object

Rizdania, S.T., M.Kom.

Class

- **Class** adalah prototype atau template dari objek.
- Sebuah **class** mempunyai anggota yang terdiri dari **atribut** dan **method**.
- **Atribut** adalah semua field identitas yang kita berikan pada suatu class, exp :
  - Class Mahasiswa : - Nim , nama , alamat.
- **Method** dapat kita artikan sebagai semua fungsi ataupun prosedur yang merupakan perilaku (behaviour) dari suatu class.

Contoh


NamaClass
namaAtribut1: tipeData
namaAtribut2: tipeData
namaMethod1(): tipeData
namaMethod2(): tipeData
Mahasiswa
+nim: int
+nama: String
+alamat: String
+kelas: String
+telepon: int
+tampilBiodata(): void
nama class
daftar atribut
daftar method

Aturan penulisan class

- Berupa kata benda,
- Diawali dengan **HURUF BESAR,**
- Jika terdiri dari lebih dari 1 kata, maka antar kata satu dengan kata yang lain **disambungkan**, dan tiap huruf awal dari tiap kata menggunakan **HURUF BESAR**.
- Contoh: Mahasiswa, SepedaDemo

Implementasi Class

› Untuk membuat suatu Class, digunakan kata kunci class dan diikuti dengan nama Class yang akan dibuat, exp:

```
<modifier> class <nama_class> {

//deklarasi atribut dan method

}
```

```
public class Mahasiswa{


}
```

Aturan penulisan atribut

- Aturan penulisan atribut adalah sebagai berikut:
- Berupa kata benda,
- Diawali dengan **HURUF KECIL**,
- Jika terdiri dari 2 atau lebih kata, kata pertama diawali **HURUF KECIL** sedangkan kata selanjutnya diawali **HURUF BESAR**. Dan antar kata disambung tidak (dipisah).
- Contoh: nama, kelas, programStudi, tampilNilaiSiswa

Deklarasi Atribut

› Untuk melakukan pendeklarasian atribut dapat dilakukan dengan sintaks sebagai berikut :

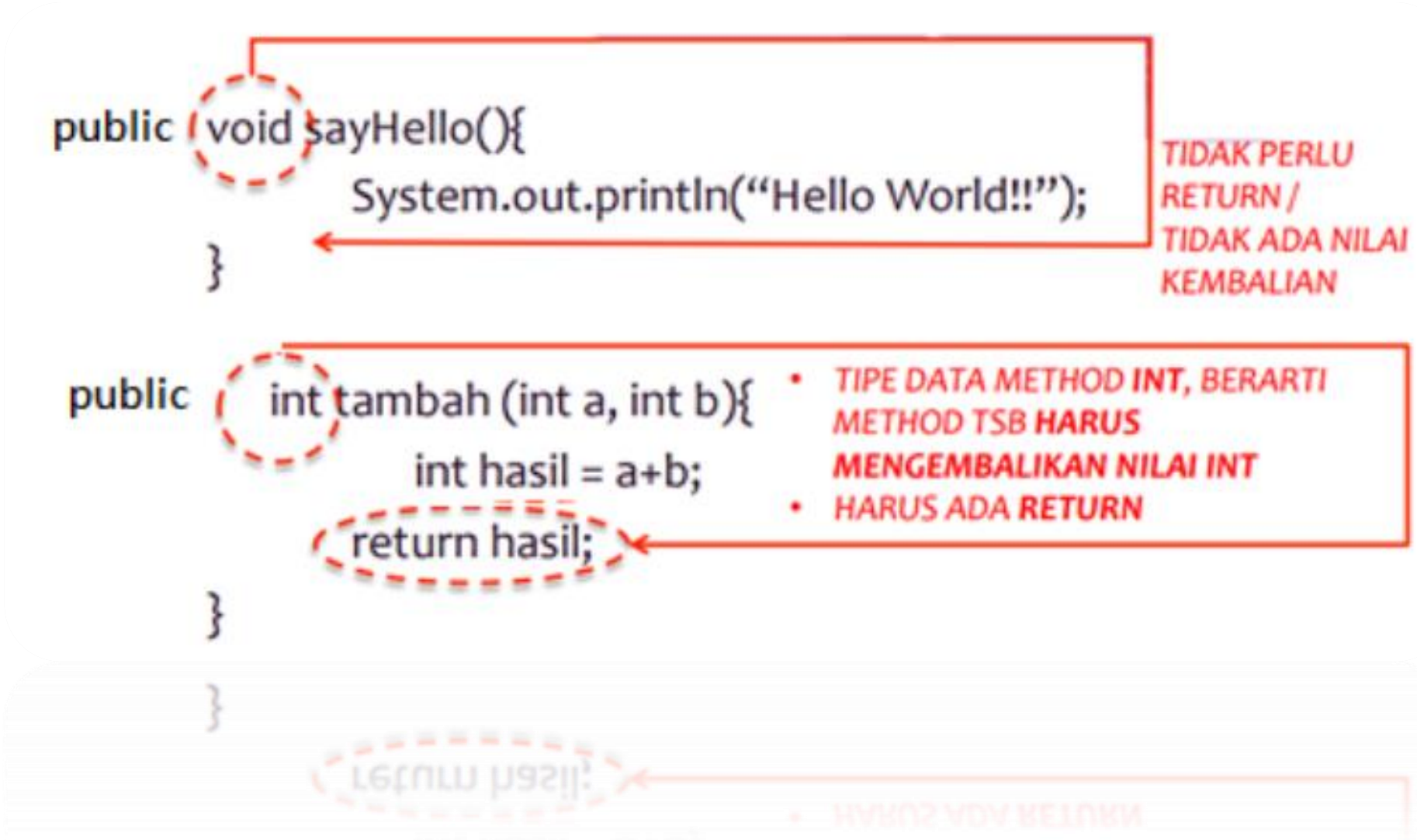
```
<modifier> <tipe> <nama_atribut> ;
```

› Exp :

```
public int nim;
public String nama;
public String programStudi;
```

Tipe Method

- Method dengan tipe data void, berarti tidak memiliki nilai balik, berarti tidak memerlukan kata kunci return di dalamnya.
- Method dengan tipe data bukan void, berarti memerlukan suatu nilai balik, yaitu harus memerlukan return di dalamnya


public void sayHello(){
System.out.println("Hello World!!");
}
TIDAK PERLU RETURN / TIDAK ADA NILAI KEMBALIAN
public int tambah (int a, int b){
int hasil = a+b;
return hasil;
}
}
TIPE DATA METHOD INT, BERARTI METHOD TSB HARUS MENGEMBALIKAN NILAI INT
HARUS ADA RETURN

Method

› Deklarasi method dapat dilakukan dengan sintaks sebagai berikut :

```
<modifier> <tipe_data> <nama_metode> ([daftar_argumen])

{

    //statement

}
```

Aturan penulisan method

- Aturan penulisan method adalah sebagai berikut:
- Berupa kata kerja,
- Diawali dengan **HURUF KECIL**,
- Jika terdiri dari 2 atau lebih kata, kata pertama diawali **HURUF KECIL** sedangkan kata selanjutnya diawali **HURUF BESAR**. Dan antar kata disambung tidak (dipisah).
- Contoh: tambahKecepatan, cetak, tampilBiodataSiswa

Deklarasi method

```
public void tampil(){
System.out.println (“Hallo PBO!!”);
}

public int tambah(int a, int b){
hasil=a+b;
return hasil;
}
```

Object

› Object adalah instansiasi dari sebuah class , misal:

```
NamaClass namaObject = new NamaClass();
```

› Mahasiswa mhs = new Mahasiswa();

Implementasi Object

```
public class Mahasiswa{
public int nim;
public int nama;
public int alamat;

        public void biodataMahasiswa(){
        System.out.println(“Biodata Mahasiswa”);
        System.out.println(“Nim          :”+nim);
        System.out.println(“Nama         :”+nama);
        System.out.println(“Alamat :”+alamat);
        }
}
```

```
public class TampilMahasiswa{

        public static void main(String[]args){
        Mahasiswa mhs = new Mahasiswa();
        mhs.nim = 1;
        mhs.nama = “Very Sugiarto”;
        mhs.alamat = “Malang”;
        mhs.biodataMahasiswa();

        }
}
```

Try – catch

- Untuk menangani ***error*** di Java, digunakan sebuah statement yang bernama *try - catch*.

- Statement tersebut digunakan untuk mengurung eksekusi yang menampilkan *error* dan dapat membuat program tetap berjalan tanpa dihentikan secara langsung.

- *Error* yang ditangani oleh *try – catch* biasa disebut dengan ***exception***.

Exception

- Exception adalah sebuah alur peristiwa yang menjalankan proses pada program, pristiwa tersebut bisa berupa kesalahan atau error pada program yang kita buat, error tersebut bisa terjadi karena beberpa faktor, diantaranya: kesalahan input, jenis format data yang dimasukan salah, peggunaan array yang melebihi batas, Dll.
- Ada banyak sekali jenis Exception yang bisa kita tangkap menggunakan fungsi *try-catch*, salah satunya adalah **ArrayIndexOutOfBoundsException**, exception ini menandakan bahwa jumlah array yang kita inputkan melebihi batas, pesan tersebut akan muncul saat aplikasi dijalankan yang akan menyebabkan terhentinya program tersebut.

Statement try-catch

- Bagaimana cara kita mengetahui jenis error atau exception apa yang muncul, pada contoh berikut ini kita akan membuat program sederhana, dimana program tersebut sengaja kita buat menjadi error atau terjadi kesalahan.
- Contoh 1:

```
public class TanpaTC {

    public static void main(String[] args) {

        int angka = 7;
        int hasil = angka/0;
        System.out.println(hasil);

    }

}
```

Statement try-catch (contd.)

- Program tersebut akan menghasilkan, output berupa kesalahan seperti ini:

```
run:
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at tanpatc.TanpaTC.main(TanpaTC.java:17)
C:\Users\Rizdania\AppData\Local\NetBeans\Cache\8.1\executor-snippets\run.xml:53: Java returned: 1
BUILD FAILED (total time: 0 seconds)
```

- Jenis Exception yang tampil pada program tersebut adalah *ArithmeticException*, error tersebut terjadi karena ada pembagian 0 (nol), jika hal ini terjadi, user akan kebingungan dengan error tersebut serta menyebabkan force close.

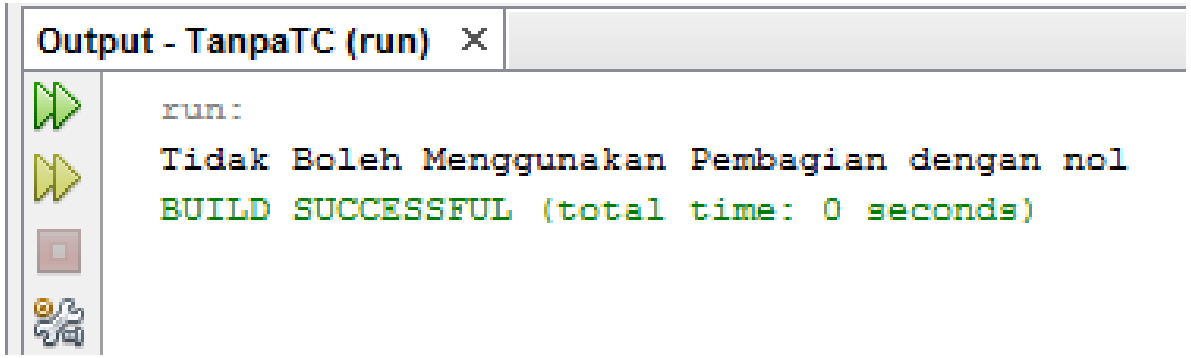
› Untuk mengatasi masalah tersebut, kita dapat menggunakan statement try-catch, seperti pada contoh berikut ini:

```
public class TanpaTC {

    public static void main(String[] args) {
        // TODO code application logic here
        try{
            // pernyataan yang berpotensi mengakibatkan Exception
           int angka = 7;
           int hasil = angka/0;
           System.out.println(hasil);
        }catch(ArithmeticException ex){
            // pernyataan disini akan di eksekusi jika terjadi Exception
            System.out.println("Tidak Boleh Menggunakan Pembagian dengan nol");
        }

    }

}
```



Contoh 2:

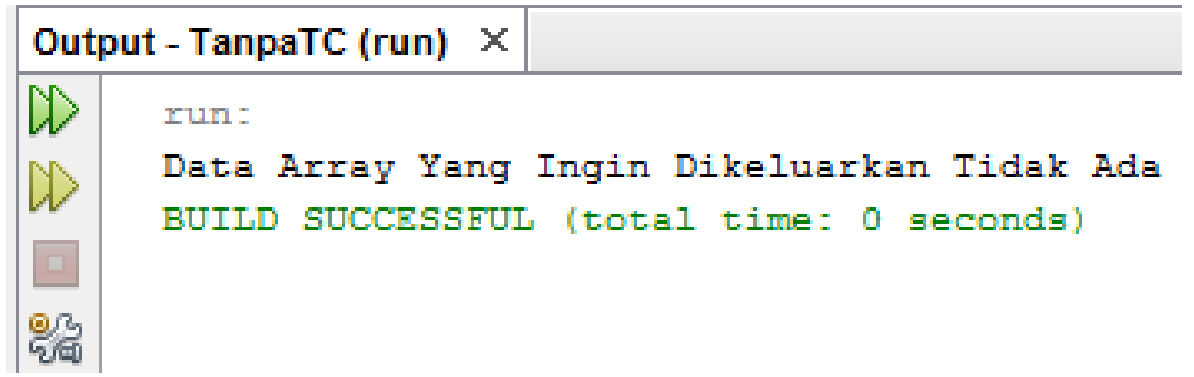
```
public class TanpaTC {

    public static void main(String[] args) {

        try{
          // pernyataan yang berpotensi mengakibatkan Exception
         String[] siswa = new String[2];
         siswa[0] = "Wildan";
         siswa[1] = "Ferdi";
         siswa[2] = "Taufiq";
         System.out.println(siswa[4]);
        }catch(ArrayIndexOutOfBoundsException ex){
            // pernyataan disini akan di eksekusi jika terjadi Exception
            System.out.println("Data Array Yang Ingin Dikeluarkan Tidak Ada");
        }

    }

}
```



Multiple Catch

- Dalam bahasa pemrograman java, kita dapat menggunakan catch lebih dari satu, untuk menangkap jenis exception yang berbeda pada pernyataan didalam try.
- Pada contoh berikut ini, kita akan menggabungkan kedua program yang sebelumnya sudah kita buat menjadi satu, didalam program tersebut kita akan menangkap 2 jenis exception yang berbeda,
  yaitu *ArithmeticException* dan *ArrayIndexOutOfBoundsException*.

Contoh:

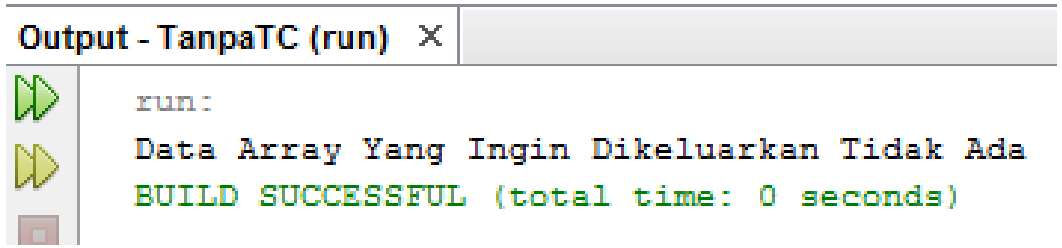
```
public class TanpaTC {

    public static void main(String[] args) {

        try{
        String[] siswa = new String[2];
        siswa[0] = "Wildan";
        siswa[1] = "Ferdi";
        siswa[2] = "Taufiq";
        System.out.println(siswa[4]);
        //====================================
        int angka = 7;
        int hasil = angka/0;
        System.out.println(hasil);
      }catch(ArrayIndexOutOfBoundsException ex){
          System.out.println("Data Array Yang Ingin Dikeluarkan Tidak Ada");
      }catch(ArithmeticException ex2){
          System.out.println("Tidak Boleh Menggunakan Pembagian dengan nol");
      }

    }

}
```



Finally

- Statement *finally* digunakan untuk mengeksekusi kode program jika terjadi exception atau tidak terjadi exception, jadi blok kode didalamnya akan terus di eksekusi pada kondisi apapun.

```
public class TanpaTC {

    public static void main(String[] args) {

         try{
        // pernyataan yang berpotensi mengakibatkan Exception
       int angka = 10;
       int hasil = angka/0;
       System.out.println(hasil);
    }catch(ArithmeticException ex){
        // pernyataan disini akan di eksekusi jika terjadi Exception
        System.out.println("Tidak Boleh Menggunakan Pembagian dengan nol");
    }finally{
        /*
        Pernyataan disini akan di eksekusi jika terjadi Exception
        Ataupun tidak terjadi Exception
        */
        System.out.println("Program Diakhiri");
    }
    }
}
```

```
Output - TanpaTC (run)
run:
Tidak Boleh Menggunakan Pembagian dengan nol
Program Diakhiri
BUILD SUCCESSFUL (total time: 0 seconds)
```

TERIMA KASIH