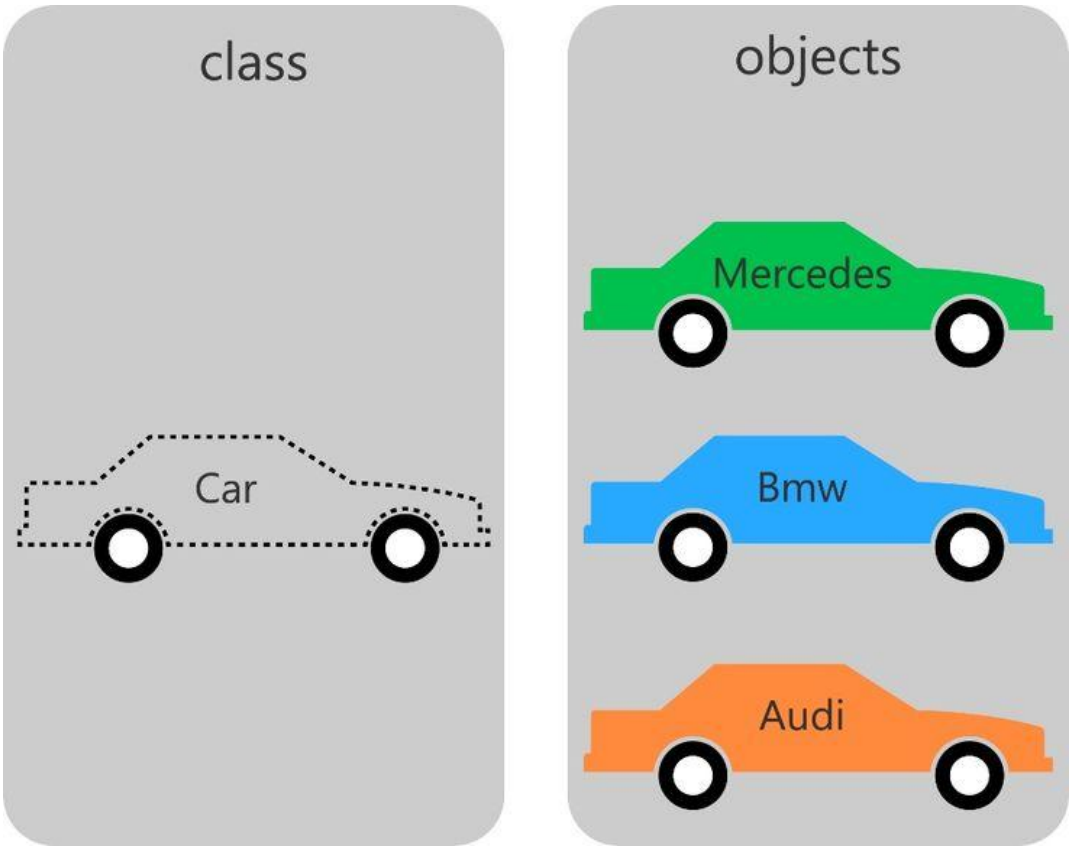
Modul 1

Kelas & Objek

Praktikum Pemrograman Berorientasi Objek 2025/2026



Universitas Atma Jaya Yogyakarta

Fakultas Teknologi Industri

Program Studi Informatika

A. Tujuan

1. Mahasiswa memahami konsep kelas dan objek.
2. Mahasiswa mampu mengimplementasikan konsep kelas dan objek ke dalam Bahasa pemrograman Java.

B. Dasar Teori

Segala sesuatu adalah **Objek**. Objek-objek yang sering kita temui di kehidupan sehari-hari tentu dapat dianggap juga sebagai sebuah objek. Objek memiliki 2 komponen, yaitu:

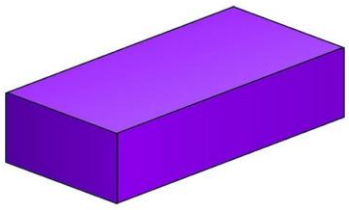
1. **Atribut/Variabel/Field**

   Sebuah variabel yang melekat pada sebuah objek yang digunakan untuk menyimpan sebuah nilai/value/data yang dimiliki oleh sebuah objek (implementasi dari atribut).

2. **Method/Operasi**

   Merupakan perilaku yang bisa dilakukan oleh objek tersebut. Implementasi dari perilaku objek adalah method (fungsi atau prosedur) yang digunakan untuk mengakses atau memanipulasi atribut sehingga menghasilkan suatu keluaran yang diharapkan.

Contoh-contoh Objek:

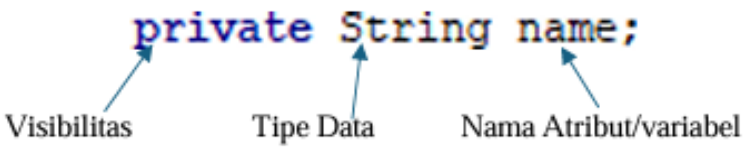
	**Nama Objek**: Balok **Atribut**: • Panjang: 10 • Tinggi: 5 • Lebar: 5 • Warna: Unggu **Method**: • Menghitung Volume • Menghitung Luas Permukaan Balok
	**Nama Objek:** Mahasiswa **Atribut:** • Nama: Gabriel • NIM: 123456789 • Jurusan: Informatika • Semester: 6 • IPK: 4 **Method:** • Mengambil Mata Kuliah • Mengumpulkan Tugas

Keterangan: Panjang → atribut, 10 → value

Objek-objek yang sama dapat dikumpulkan menjadi sebuah **Kelas**. Semua objek dalam suatu kelas memiliki atribut dan operasi yang sama, tetapi nilai atributnya dapat berbeda. Suatu objek merupakan instance sebuah kelas. Kelas merupakan sebuah **template** yang digunakan untuk membentuk sebuah objek dengan karakteristik (atribut dan operasi) yang sama. Sebuah kelas memiliki beberapa komponen pembentuk utama, yaitu:

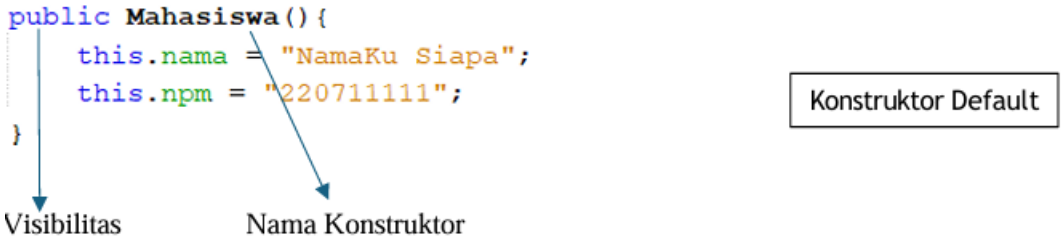
1. **Atribut**

   Atribut merupakan tempat untuk menyimpan sebuah nilai yang dimiliki oleh sebuah objek. Pada saat sebuah objek dibuat dari sebuah kelas, maka atribut yang ada di kelas tersebut juga akan melekat pada objek yang diciptakan. Dalam implementasinya, atribut mungkin dapat juga berupa objek lain, atau rujukan ke objek lain. Dalam bahasa pemrograman Java, pendeklarasian atribut dapat dituliskan sebagai berikut:



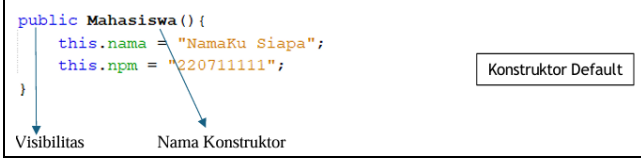
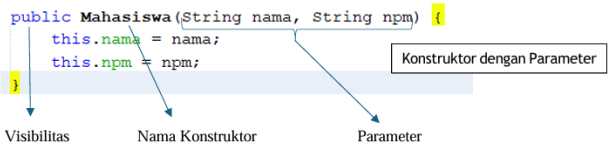
2. **Konstruktor**

   Konstruktor merupakan method yang berfungsi untuk membuat objek dan menginisialisasikan nilai-nilai atribut dari objek yang terbentuk. Tujuan dari penggunaan konstruktor adalah untuk memastikan bahwa objek yang dihasilkan oleh kelas berada dalam sebuah keadaan yang valid. Suatu kelas dapat berisi lebih dari 1 konstruktor, dan karakteristik utamanya adalah tidak memiliki nilai balikan, dan nama konstruktor pasti sama dengan nama kelasnya. Dalam bahasa pemrograman Java,



   konstruktor dapat dituliskan sebagai berikut**:**

Perbedaan Konstruktor Default dan Konstruktor Dengan Parameter:

Konstruktor Default	Konstruktor Dengan Parameter
Konstruktor tanpa parameter yang dipanggil saat objek dibuat tanpa mengirim nilai apa pun.	Konstruktor yang menerima nilai dari luar saat objek dibuat.
Ciri-ciri: - Tidak punya parameter - Biasanya untuk memberi nilai awal (default value)	Ciri-ciri: - Memiliki parameter - Bisa langsung mengisi atribut sesuai input
Contoh: Pada konstruktor default diatas kita tidak memasukkan parameter apapun oleh karena itu setiap kali objek dibuat maka atribut nama akan selalu bernilai “Namaku Siapa” dan atribut npm akan selalu bernilai “220711111” 	Contoh: Pada konstruktor dengan parameter dibawah bisa dilihat bahwa atribut nama dan juga npm akan di set sesuai dengan yang ada di parameter sehingga ketika kita ingin membuat objek mahasiswa kita bisa mengubah nama dan npm nya menjadi seperti yang kita mau. 

3. **Method**

   Perilaku yang bisa dilakukan oleh objek tersebut. Implementasi dari perilaku adalah method yang digunakan untuk mengakses atau memanipulasi atribut sehingga menghasilkan suatu keluaran yang diharapkan. Method dapat berupa prosedur dan fungsi. Untuk fungsi, dapat memiliki nilai balikan berupa tipe data dasar & bentukan (int, double, String, vector, dll) dan dapat berupa kelas lain. Terdapat beberapa jenis method, yaitu:

   1. **Aksesor/Accessor/Getter**

      Sebuah Method disebut dengan method aksesor jika fungsi dari method tersebut digunakan untuk mengembalikan nilai dari atribut sebuah objek. Tipe balikan

method aksesor biasanya mengikuti tipe data atribut yang nilainya akan dikembalikan (mendapatkan value dari suatu atribut). Contohnya:

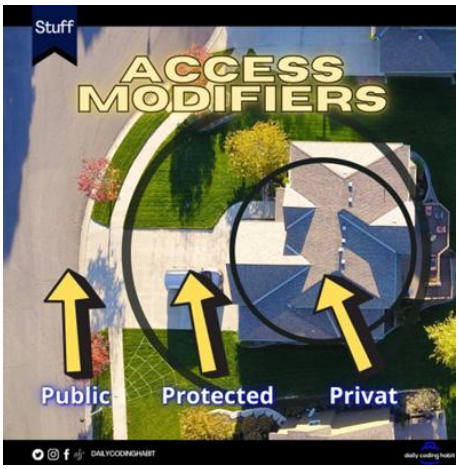
```
public String getNama() {
    return nama;
}
//method getNama() akan mengembalikan nilai atribut nama
```

2. **Mutator/Setter**

Sebuah method disebut dengan method mutator jika fungsi dari method tersebut adalah untuk mengubah nilai/value dari atribut tertentu. Method mutator biasanya tidak mengembalikan nilai apapun/void. Contohnya:

```
public void setNama(String nama) {
    this.nama = nama;
}
//method setNama(...) akan mengganti value dari variabel
//nama dengan value dari parameter inputan String nama
```

4. **Visibilitas**



Visibilitas atau Aksesibilitas bisa juga dibilang sebagai “batas” akses anggotanya, terdapat 3 tipe visibilitas yang ada, yaitu:

No	Visibilitas	Simbol	Deskripsi
1.	Private	-	TIDAK dapat diakses oleh kelas lain.
2.	Protected	#	Dapat diakses oleh kelas turunan.
3.	Public	+	Dapat diakses oleh kelas lain.

5. **Enkapsulasi**

Pendeklarasian variabel adalah perwujudan dari sebuah **enkapsulasi**. Enkapsulasi merupakan salah satu prinsip dasar dari Pemrograman Berbasis Objek (PBO) yang bertujuan untuk **membungkus atribut** yang tersedia pada kelas. Pembungkusan ini dilakukan agar sebuah atribut tidak dapat sembarang diakses oleh kelas lainnya. Akses terhadap atribut dapat dilakukan melalui methodnya.

Sesuai dengan ilustrasi di atas, pemberian hak akses **private** terhadap sebuah field mengasumsikan kita sudah mengamankan field tersebut dari akses yang tidak diizinkan. Sementara itu jika kita memberikan akses **public** kepada sebuah field maka **kita tidak menyembunyikan field** dari akses yang tidak diizinkan. Itulah hal yang dimaksud pada cover modul ini, jika kita menggunakan visibilitas public pada sebuah field maka field tersebut menjadi “dapat diakses oleh sembarang kelas” atau dengan kata lain melanggar sebuah konsep enkapsulasi.

Selain komponen utama sebuah kelas, kalian juga perlu mengetahui jenis-jenis variable, terdapat 2 jenis Variable:

1. **Instance Variable/Variabel Objek**

   Setiap objek memiliki salinan sendiri dan salinan tersebut memiliki nilai masing-masing. Perhatikan contoh di bawah, dapat dilihat bahwa setiap objek memiliki value/nilai atributnya masing-masing.

   Contoh:

   Kelas → Mahasiswa

   Objek → mahasiswa1 → nama = “Eric”, umur = 20

   mahasiswa2 → nama =”Fajar”, umur = 21

2. **Class Variable/Variabel Kelas**

   Suatu variabel yang dimiliki pada suatu kelas dan variable tersebut akan digunakan Bersama-sama oleh semua objek dari kelas tersebut. Menggunakan keyword **static**.

   Contoh:

   Kelas → Mahasiswa → public static int jumlahObjekCreated

   Objek → mahasiswa1 → nama = “Eric”, umur = 20

   mahasiswa2 → nama =”Fajar”, umur = 21

   Terdapat class variable bernama **jumlahObjekCreated** nanti akan menampilkan value 2 (karena kita membuat 2 Objek). Value dari variable **jumlahObjekCreated** dapat diakses oleh semua objek yang berada pada kelas Mahasiswa.

Konstanta adalah sesuatu yang nilainya tetap, tidak bisa diubah dan nilainya akan selalu sama. Diberikan kepada variabel yang nilainya sudah pasti. Menggunakan keyword **final**, karena nilai yang sudah terdeklarasikan tidak akan bisa diubah lagi. Umumnya penulisan untuk variabel konstanta berisi huruf besar semuanya.

Contoh pendeklarasian:

```
public class Lingkaran {
    private final double PHI = 3.14; //pendeklarasian konstanta
    private double jari_jari; //pendeklarasian atribut

    public double luasLingkaran(){
        return PHI * jari_jari * jari_jari;
    }
}
```

Bagaimana cara menggunakan konstanta **PHI** yang ada di dalam kelas lingkaran untuk digunakan pada kelas lainnya?

Sebagai contoh, kita buat kelas Tabung. Kemudian kita akan menggunakan class variable/variabel kelas untuk dapat diakses oleh kelas lain, dengan keyword **static**

Contoh pendeklarasian:

```
public class Lingkaran {
    public static final double PHI = 3.14; //pendeklarasian static konstanta
    private double jari_jari; //pendeklarasian atribut

    public double luasLingkaran(){
        return PHI * jari_jari * jari_jari;
    }
}
```

Dengan menambahkan keyword **static** pada **PHI**, maka sekarang PHI dapat digunakan oleh kelas lain dengan catatan visibilitas **public**.

Contoh pendeklarasian:

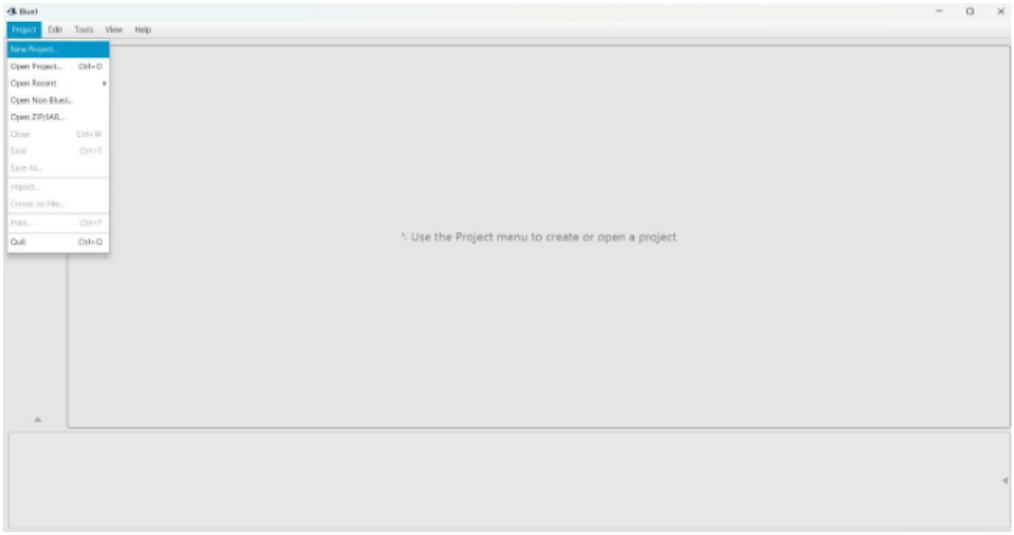
```
public class Tabung {
    private double jari_jari; // pendeklarasian atribut
    private double tinggi; //pendeklarasian atribut

    public double volumeTabung() {
        return Lingkaran.PHI * jari_jari * jari_jari;
        //memanggil variabel konstanta dengan nama KELAS
        //(dot) namaVariabelKonstantanya
    }
}
```

GUIDED 1.1 – BLUE JAVA

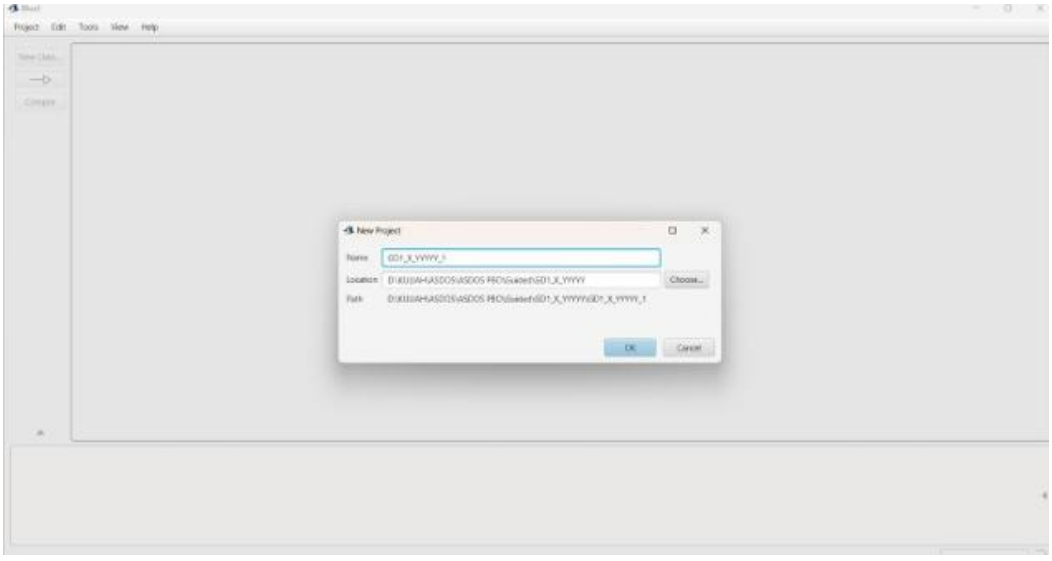
Guided pertama ini, kita akan membuat sebuah kelas bernama Segitiga dengan menggunakan salah satu tools untuk belajar bahasa pemrograman java, yaitu BlueJ. Secara umum sebuah segitiga mempunyai 2 buah atribut, yaitu alas dan tinggi. Lalu kita akan mencoba membuat 2 method yaitu menghitung luas dan mencari pytagoras segitiga tersebut.

LANGKAH – LANGKAH

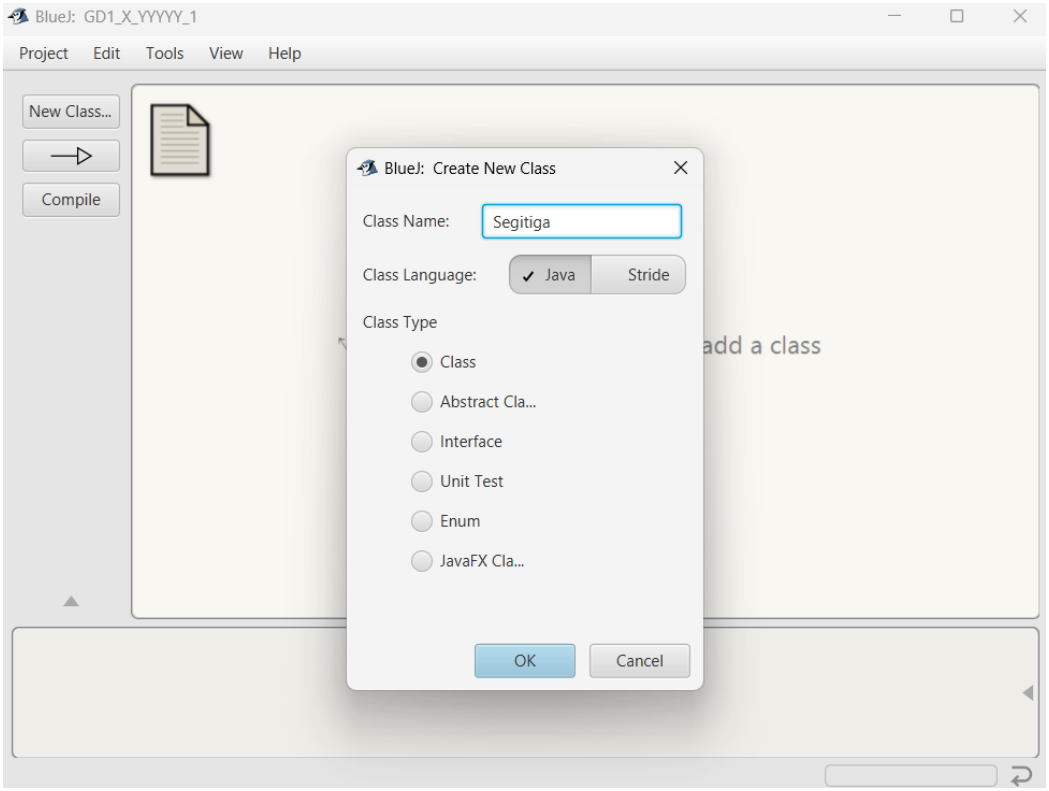
1. Buka aplikasi BlueJ kalian. Kemudian pilih Project → New Project



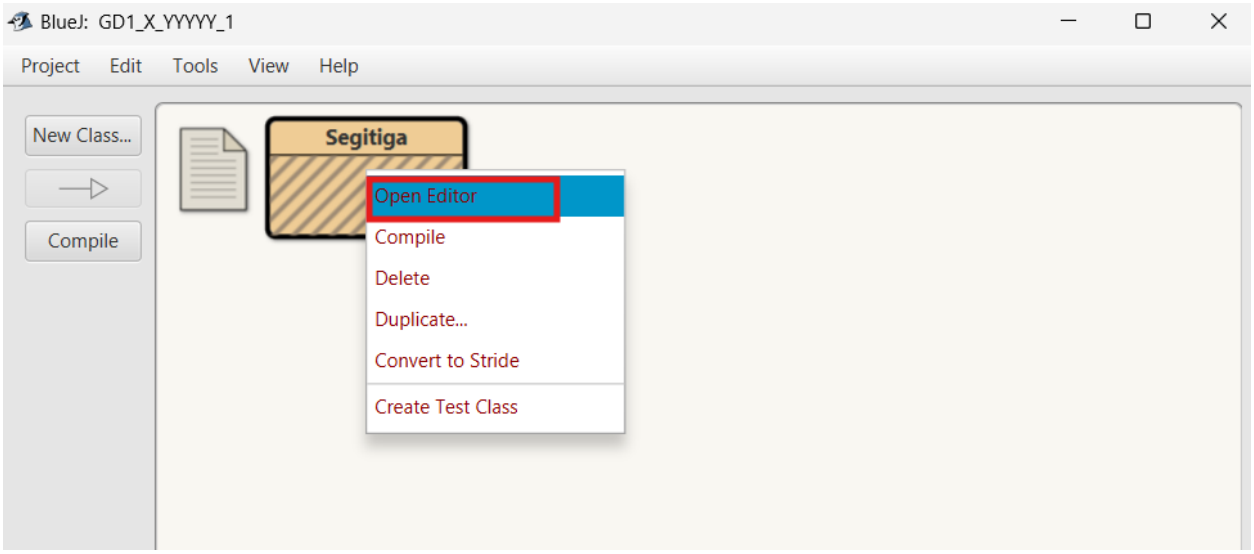
2. Isi Name dengan format **GD1_X_YYYYY_1 (X = Kelas, YYYYY = 5 Digit Belakang NPM)**, lalu klik OK. Jendela BlueJ baru akan terbuka.



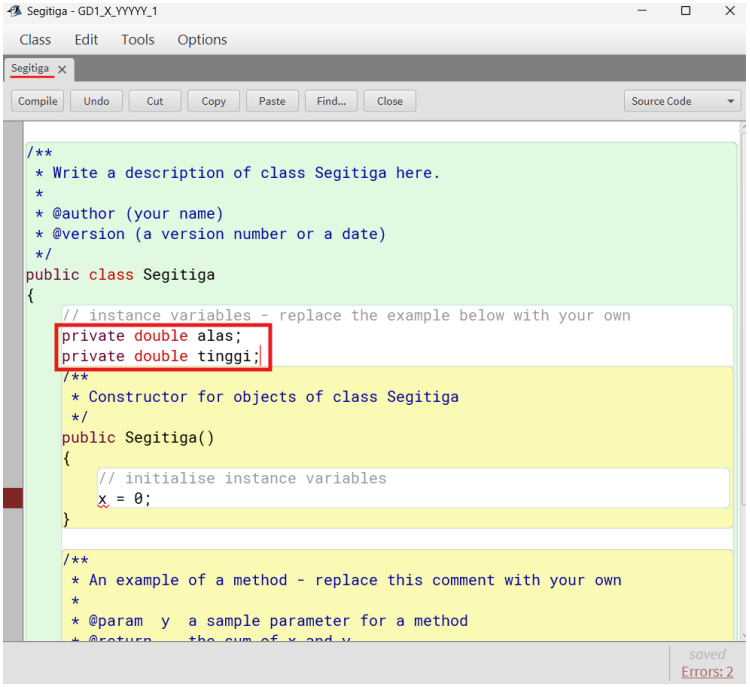
3. Sekarang, kita akan membuat sebuah kelas yang bernama Segitiga, dengan cara klik tombol **New Class**, isi bagian **Class Name** dengan nama Segitiga, lalu klik OK.



4. Sekarang, kita akan memulai mengedit code milik kelas Segitiga dengan cara **Klik kanan** pada kelas Segitiga, Klik **Open Editor**



5. Masukkan 2 Atribut yang dimiliki oleh segitiga, yaitu **alas** dan **tinggi** yang bertipe data **double**.



Catatan:

Dalam langkah ini, akan memunculkan error yang bisa kita lihat dengan adanya 2 kotak merah yang berada di samping kiri code (pada baris x=0 & return x+y). Error tersebut muncul karena variabel yang kita pakai tidak kita lakukan pendeklarasian atribut. Maka dari itu, usahakan untuk memperhatikan penulisan code (pendeklarasian atribut, penggunaan semicolon, dll), sehingga tidak terjadi adanya syntax error yang menyebabkan program tidak dapat berjalan.

6. Kalian boleh untuk menghapus code berikut, karena code tersebut tidak akan kita pakai nantinya.

```
/**
 * An example of a method - replace this comment with your own
 *
 * @param  y  a sample parameter for a method
 * @return    the sum of x and y
 */
public int sampleMethod(int y)
{
    // put your code here
    return x + y;
}
```

7. Isilah konstruktor pertama public `Segitiga () {}` dengan nilai default, tanpa parameter (karena kita tidak meminta inputan dari user). Tambahkan konstruktor kedua yang memiliki parameter alas dan tinggi.

```
public class Segitiga
{
    // instance variables - replace the example below with your own
    private double alas;
    private double tinggi;
    /**
     * Constructor for objects of class Segitiga
     */

    // konstruktor Default
    public Segitiga()
    {
        alas = 10.2;
        tinggi = 12.4;
    }

    // konstruktor dengan parameter
    public Segitiga(double alas, double tinggi)
    {
        this.alas = alas;
        this.tinggi = tinggi;
    }
}
```

Catatan:

"**this.**" Sebagai petunjuk bahwa variable yang dimaksud adalah atribut dari kelas tersebut.

BUKAN PARAMETER INPUTAN

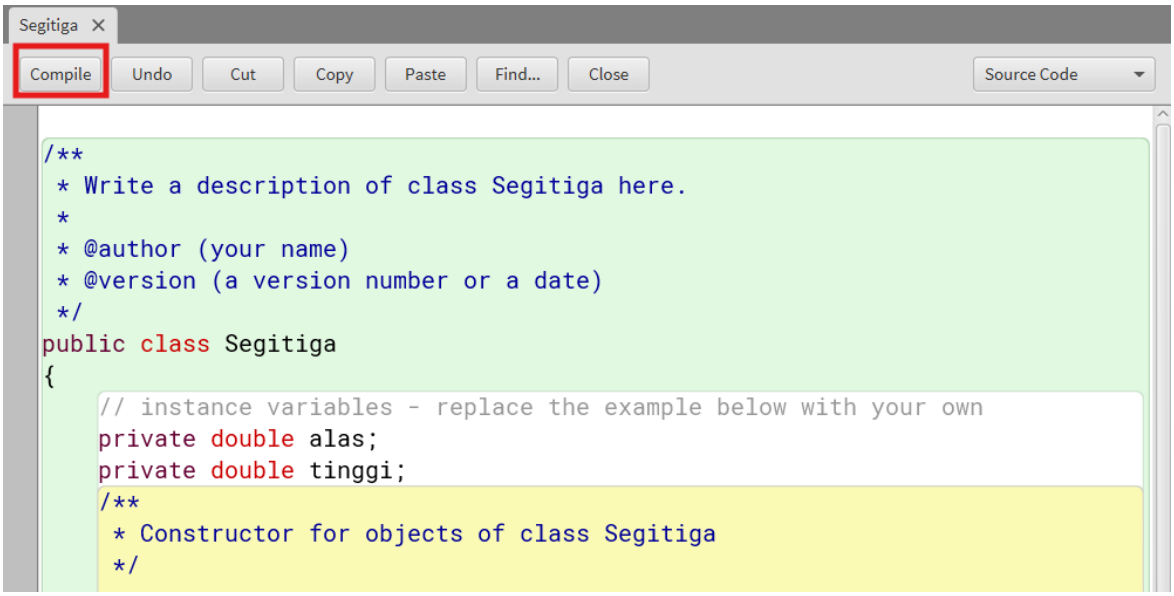
Ada beberapa cara penulisan konstruktor, yaitu

```
public Segitiga(double alas, double tinggi)
{
    this.alas = alas;
    this.tinggi = tinggi;
}
```

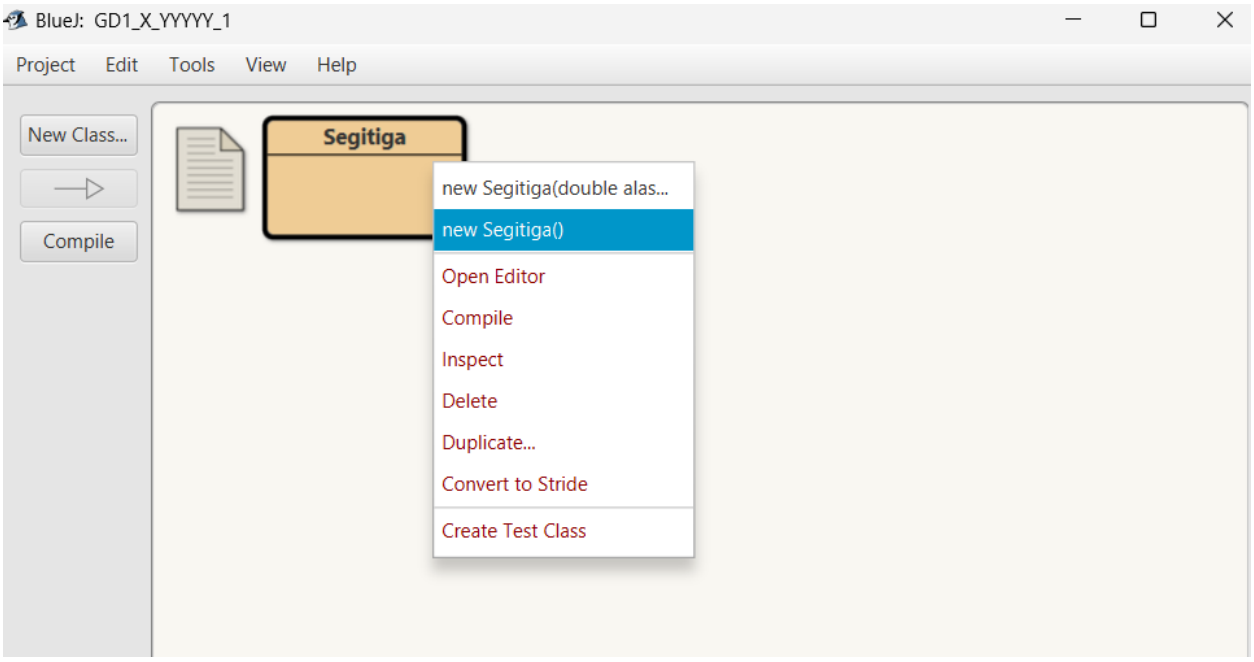
atau

```
public Segitiga(double inputanAlas, double inputanTinggi)
{
    alas = inputanAlas;
    tinggi = inputanTinggi;
}
```

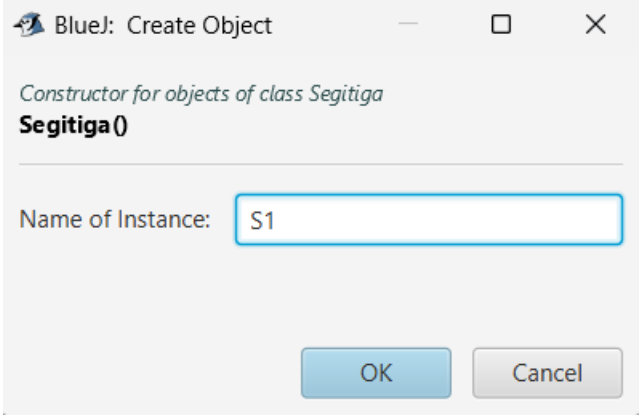
8. Setelah kalian menuliskan konstruktor default dan konstruktor dengan parameter **Klik Compile**, lalu minimize jendela editornya.



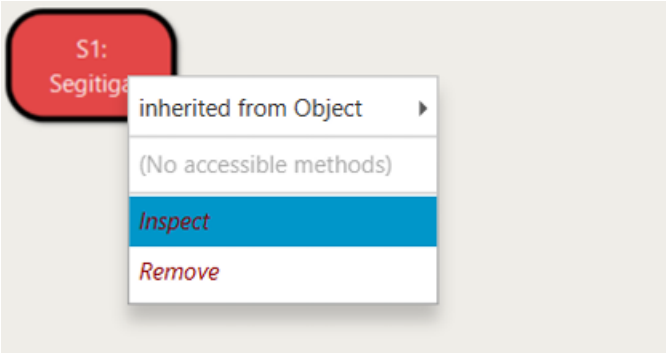
9. Setelah itu kita akan mulai untuk membuat objek dari kelas yang sudah kita buat. Caranya klik kanan di kelas Segitiga lalu pilih `New Segitiga()`



10. Beri nama objek tersebut dengan S1, klik OK

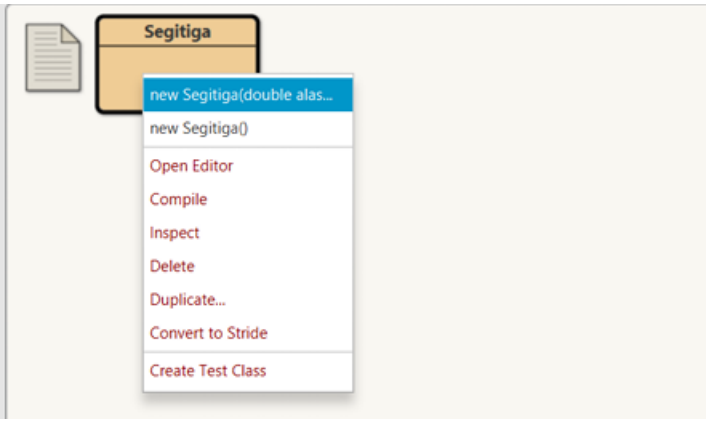


11. Lalu akan muncul kotak merah pada field Bagian bawah dengan nama S1, yang berarti kita sudah berhasil untuk membuat objek dari kelas Segitiga. Untuk kalian yang ingin melakukan pengecekkan nilai dari atribut yang ada, silahkan **Klik kanan** pada kota merah lalu klik **Inspect**.

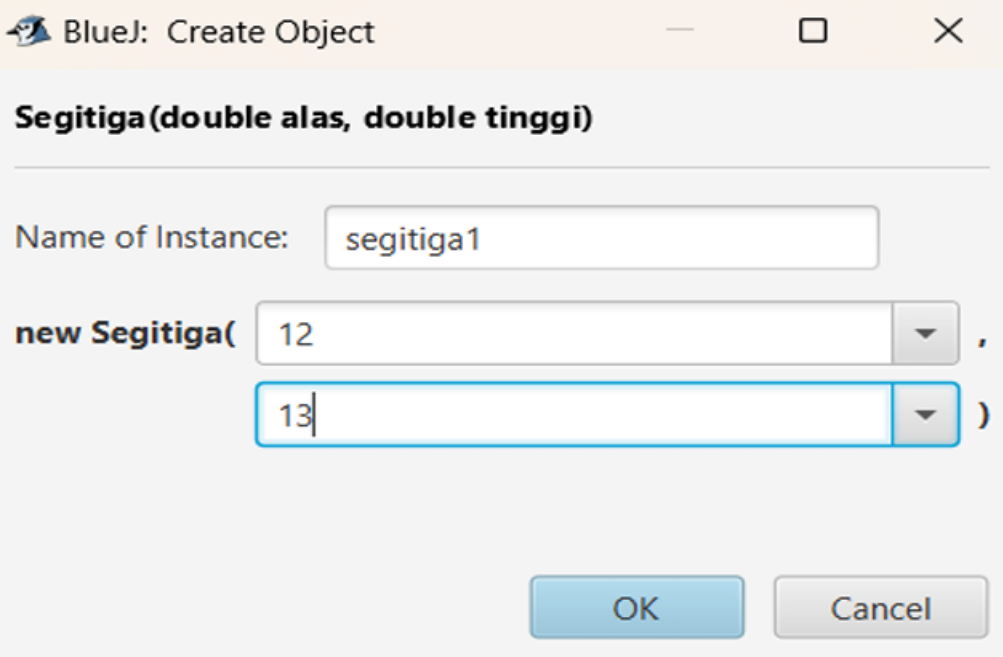


12. Lalu akan muncul sebuah kotak berwarna Merah lagi. Namun kotak merah tersebut berisi informasi tentang atribut-atribut dan nilai-nilai dari objek yang berasal dari konstruktor default yang telah kita buat.

13. Setelah kita mencoba melihat objek yang berasal dari konstruktor default, sekarang kita akan mencoba membuat objek menggunakan konstruktor kedua, yaitu konstruktor dengan parameter. Dengan cara yang sama yaitu klik kanan pada kelas Segitiga, lalu pilih `New Segitiga(double alas, double tinggi)`

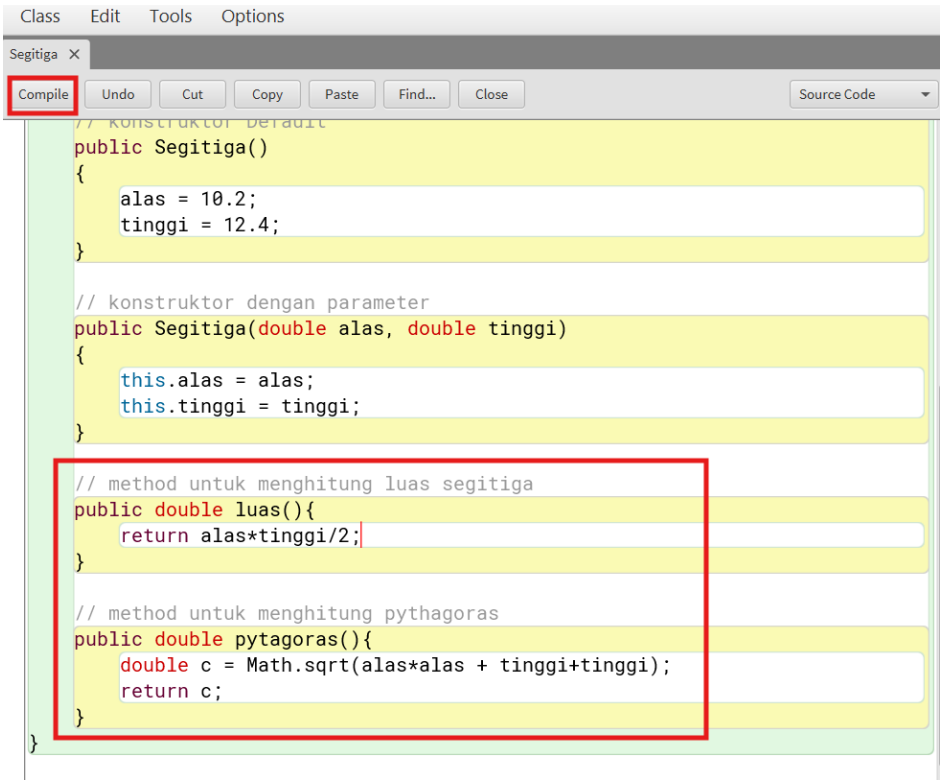


14. Sebuah jendela baru akan muncul **Name of Instance**, dan ada field yang meminta user untuk menginputkan nilai alas dan tinggi. Silahkan menginputkan alas dan tinggi sesuai dengan keinginan kalian. Jika sudah Klik OK.



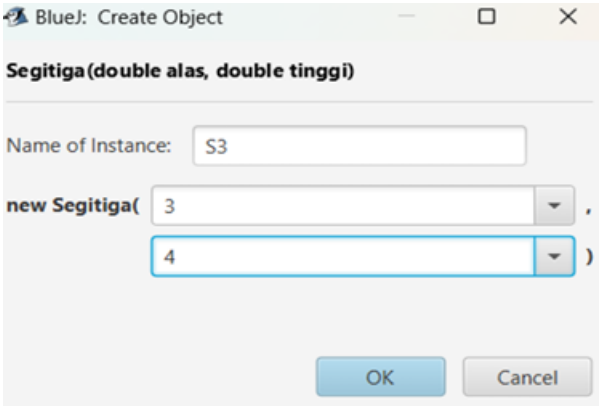
15. Lalu apabila kalian ingin melihat nilai dari atributnya silahkan lakukan cara yang sama dengan Langkah ke-11. Lalu lihatlah hasil nilai tersebut apakah nilainya sama dengan nilai yang kalian inputkan tadi.

16. Sekarang kita akan menambahkan beberapa method pada kelas Segitiga. Method yang akan kita buat adalah method untuk menghitung luas dan pytagoras dari alas dan tinggi yang ada. Buka code editor yang kalian minimize sebelumnya, lalu tambahkan code di bawah ini. Jika sudah, klik compile lagi.



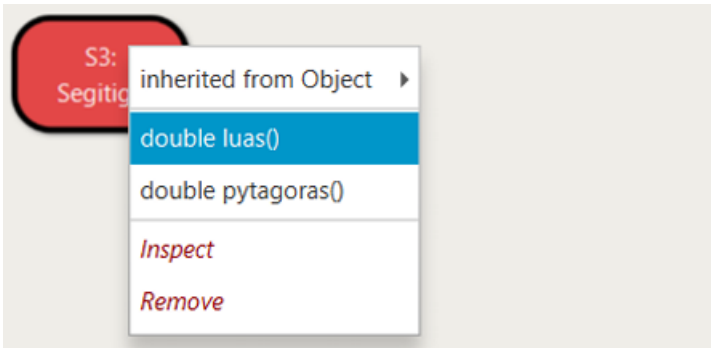
17. Kita akan mencoba memanggil method-method tersebut. Buatlah objek baru dengan nama S3, menggunakan konstruktor dengan parameter, pilihlah `New Segitiga(double alas, double tinggi)`

18. Masukkan value alas dan tinggi sesuai dengan keinginan kalian, jika sudah klik OK.

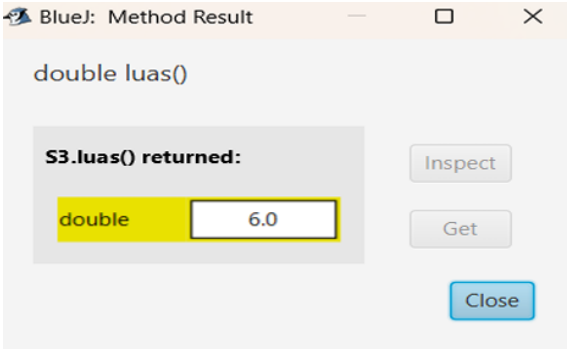


19. Jika sudah kalian Inspect atribut-atribut dan value dari atribut tersebut.

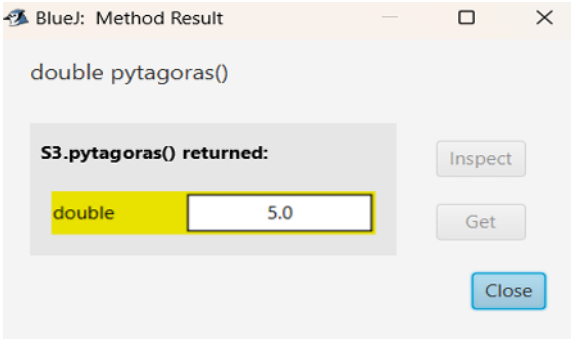
20. Sekarang kita akan mencoba melihat keluaran dari method yang telah kita buat yaitu Method `luas()` , caranya klik kanan objek kalian lalu klik `double luas()`



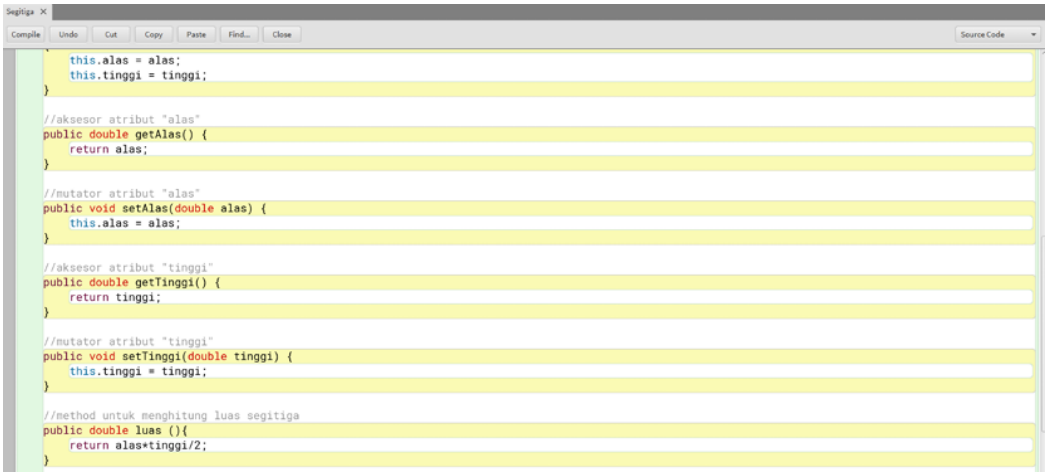
21. Lalu akan muncul jendela “Method Result” yang berisi hasil dari method `double luas()`



22. Lakukan hal yang sama dengan langkah ke-20. Cobalah untuk melihat keluaran dari method double pytagoras() yang telah kalian buat juga, maka hasilnya akan sebagai berikut.

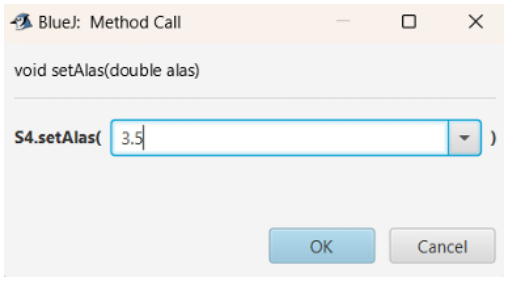
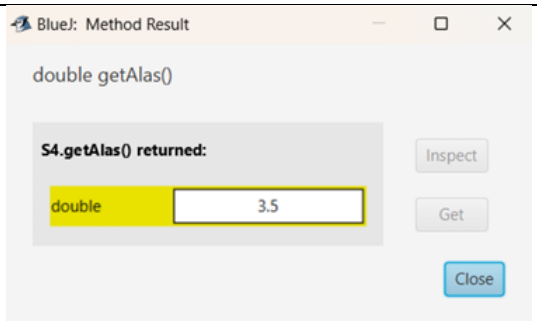


23. Setelah kita menambahkan konstruktor dan method pada kelas Segitiga. Kita juga bisa menambahkan aksesor dan mutator untuk dapat mengakses dan memanipulasi value dari masing-masing atribut.



```
        this.alas = alas;
        this.tinggi = tinggi;
    }

    //aksesor atribut "alas"
    public double getAlas() {
        return alas;
    }

    //mutator atribut "alas"
    public void setAlas(double alas) {
        this.alas = alas;
    }

    //aksesor atribut "tinggi"
    public double getTinggi() {
        return tinggi;
    }

    //mutator atribut "tinggi"
    public void setTinggi(double tinggi) {
        this.tinggi = tinggi;
    }

    //method untuk menghitung luas segitiga
    public double luas (){
        return alas*tinggi/2;
    }
```

24. Jika sudah menambahkan **Aksesor** dan **Mutator**, Compile code kalian. Kemudian buat objek baru bernama S4, dengan menggunakan konstruktor berparameter New `Segitiga(double alas, double tinggi)` dan panggil method aksesor dan mutator dari masing-masing atribut, dan juga memanggil method `luas()` dan `pytagoras()`.

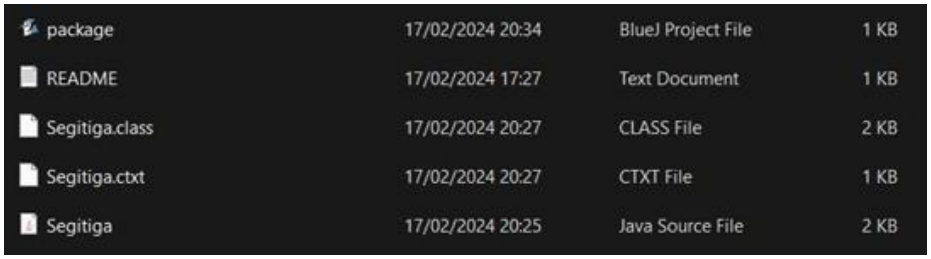
Mutator	Aksesor
	

Kemudian di Inspect

Jika kalian ingin berlatih lagi kalian bebas untuk menambah bangun datar atau objek dan kelas lainnya. Tidak perlu membuat project baru, cukup menambahkan kelas baru di project ini saja.

Ketentuan Pengumpulan Guided

1. Jika kalian mengikuti cara yang sama maka pada file yang terbentuk akan seperti pada gambar di bawah ini. **Jumlah file tergantung dari berapa jumlah kelas yang kamu buat pada 1 project! Pastikan ada .class, .ctxt, package bluej dan .java**



2. Jadikan satu file-file tersebut ke dalam satu folder yang bernama **GD1_X_YYYYY_1**
3. Kemudian zip folder tersebut dengan nama **GD1_X_YYYYY_1.zip**

   Keterangan:

   X = Kelas

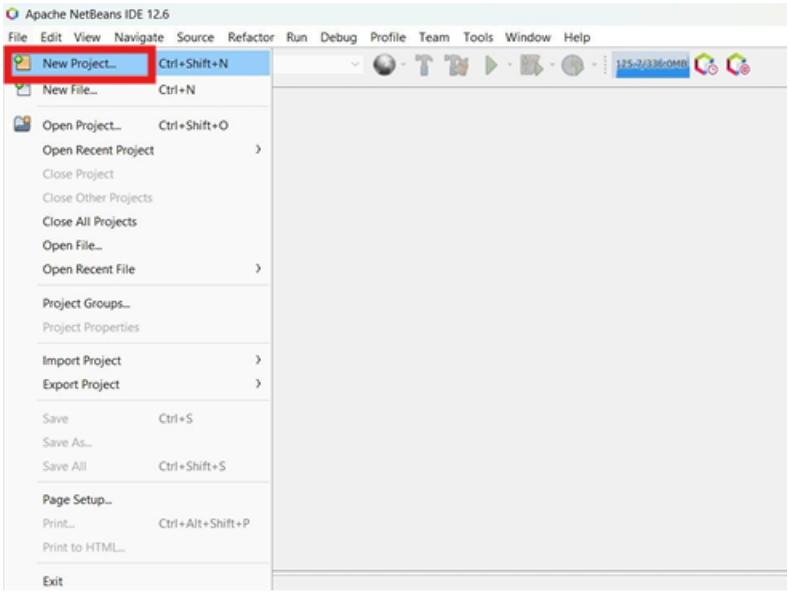
   YYYYY = 5 Digit terakhir NPM

GUIDED 1.2 – NETBEANS

Pada Guided kedua ini kita akan mengimplementasikan code yang sudah kita tuliskan pada Guided 1.1 ke dalam Netbeans.

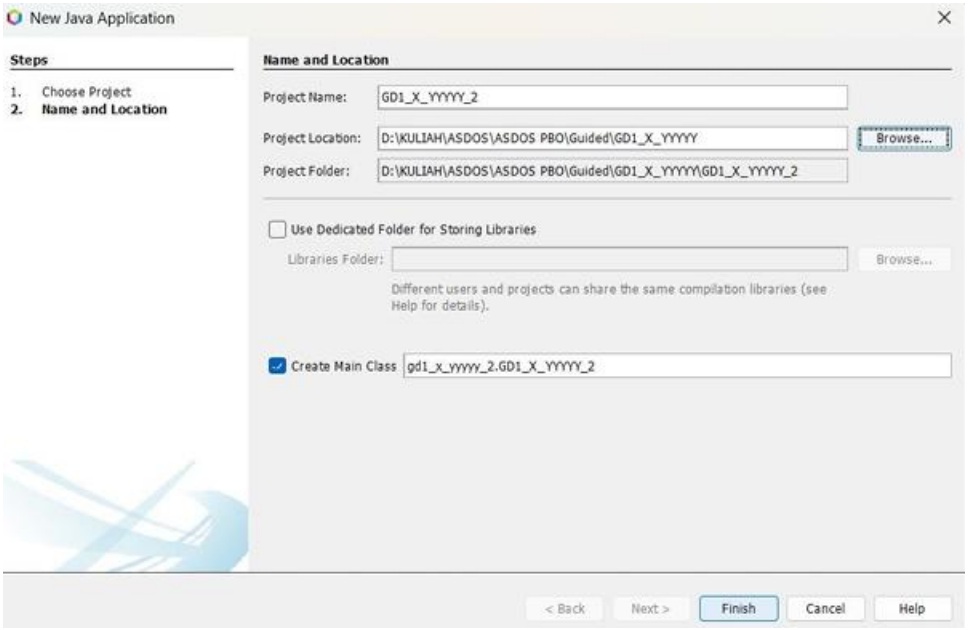
LANGKAH – LANGKAH

1. Buka NetBeans kalian, lalu klik File → New Project



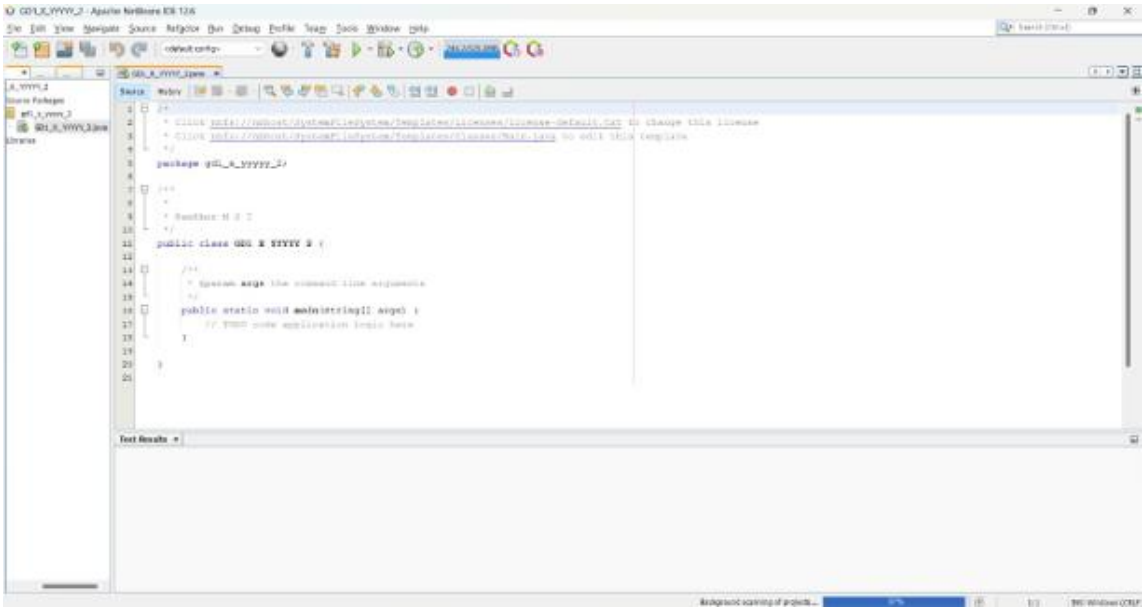
2. Kemudian pada bagian **Choose Project** kalian pilihlah Java Application, lalu klik **Next**

3. Isi Bagian **Project Name** dengan nama **GD1_X_YYYYY_2 (X = Kelas, YYYYY = 5 Digit terakhir NPM).** Jika sudah Klik Finish.

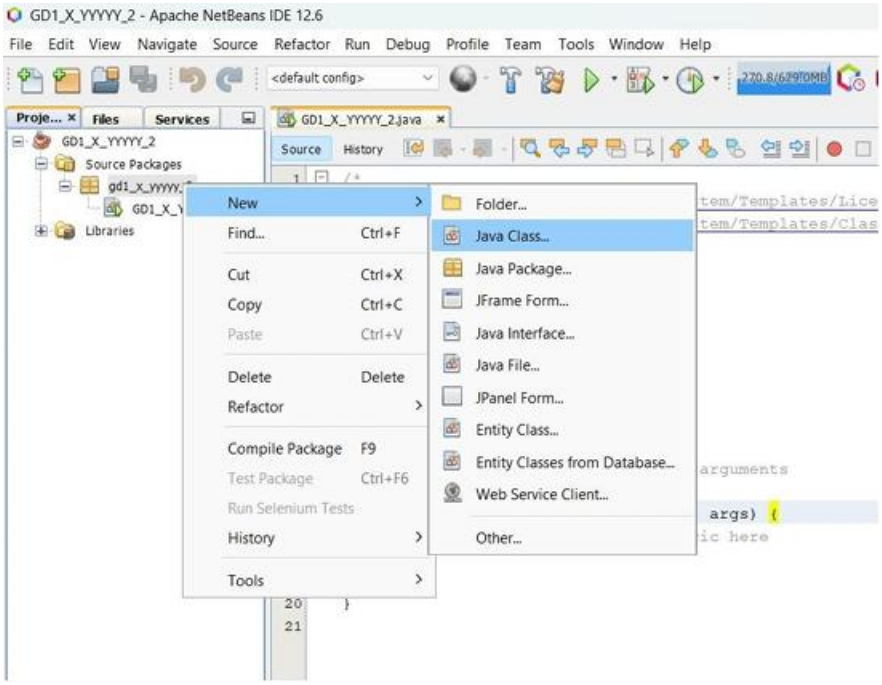


Pastikan bahwa "Create Main Class" kalian sudah dicentang. Ini bertujuan untuk kalian langsung dibuatkan main class pada saat membuat project baru.

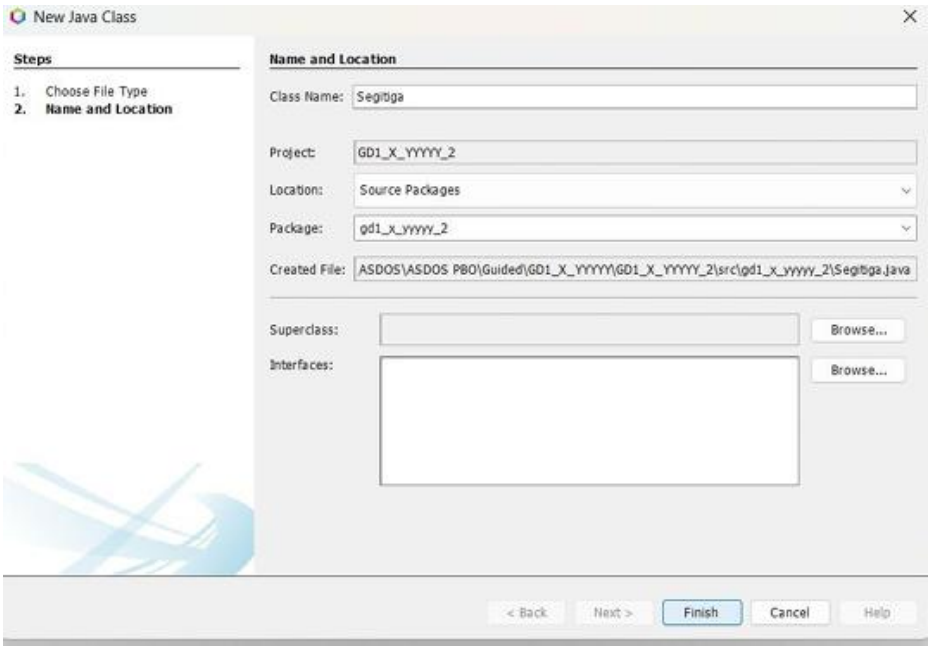
4. Kalian akan dibawa ke project baru yang kalian buat. Kurang lebihnya akan seperti ini.



5. Sekarang kita akan membuat kelas baru bernama Segitiga. Caranya Klik kanan pada **package → New → Java Class**



6. Kemudian akan muncul jendela. Isi **Class Name** dengan nama Segitiga, jika sudah klik **Finish**.



7. Maka pada bagian package di sebelah kiri akan terbentuk kelas baru dengan nama **Segitiga.java**

8. Lalu buka file **Segitiga.java** tersebut lalu ketikan code seperti di bawah ini.

```java
public class Segitiga {
    private double alas;
    private double tinggi;

    // konstruktor Default
    public Segitiga() {
        alas = 10.2;
        tinggi = 12.4;
    }

    // konstruktor dengan parameter
    public Segitiga(double alas, double tinggi) {
        this.alas = alas;
        this.tinggi = tinggi;
    }

    // aksesor atribut "alas"
    public double getAlas() {
        return alas;
    }

    // mutator atribut "alas"
    public void setAlas(double alas) {
        this.alas = alas;
    }

    // aksesor atribut "tinggi"
    public double getTinggi() {
        return tinggi;
    }

    // mutator atribut "tinggi"
    public void setTinggi(double tinggi) {
        this.tinggi = tinggi;
    }

    // method untuk menghitung luas segitiga
    public double luas() {
        return alas * tinggi / 2;
    }

    // method untuk menghitung pytagoras/sisi miring
    public double pytagoras() {
        double c = Math.sqrt(alas * alas + tinggi * tinggi);
        return c;
    }
}
```

9. Kita buka main class **(GD1_X_YYYYY_2.java)** dan tambahkan code berikut:

```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
 */
package gd1_x_yyyyy_2;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class GD1_X_YYYYY_2 {
    public static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    public static void main(String[] args) throws IOException {
        double alas;
        double tinggi;
        // Buat objek S1, menggunakan konstruktor default
        Segitiga S1 = new Segitiga();

        // Buat objek S2, menggunakan konstruktor dengan parameter
        Segitiga S2 = new Segitiga(3.5, 4.2);

    }
}
```

10. Kemudian tambahin code ini dibawahnya **(tetap di dalam main)**

```
        System.out.println("---Segitiga S1 Lama---");
        System.out.println("Alas S1     : " + S1.getAlas());
        System.out.println("Lebar S1    : " + S1.getTinggi());
        System.out.println("Pytagoras S1: " + S1.pytagoras());
        System.out.println("Luas S1     : " + S1.luas());
        System.out.println("-----------------------------\n");

        System.out.println("---Segitiga S2 Lama---");
        System.out.println("Alas S2     : " + S2.getAlas());
        System.out.println("Lebar S2    : " + S2.getTinggi());
        System.out.println("Pytagoras S2: " + S2.pytagoras());
        System.out.println("Luas S2     : " + S2.luas());
        System.out.println("-----------------------------\n");

        System.out.println("--- Input Alas dan Tinggi BARU ---");
        System.out.println("Masukkan Nilai Alas yang baru : ");
        alas = Double.parseDouble(br.readLine());

        System.out.println("Masukkan Nilai Tinggi yang baru : ");
        tinggi = Double.parseDouble(br.readLine());

        // mengubah value atribut dari objek S2 melalui Mutator/setter
        S2.setAlas(alas);
        S2.setTinggi(tinggi);

        System.out.println("---Segitiga S1 Baru---");
        System.out.println("Alas S1     : " + S1.getAlas());
        System.out.println("Lebar S1    : " + S1.getTinggi());
        System.out.println("Pytagoras S1: " + S1.pytagoras());
        System.out.println("Luas S1     : " + S1.luas());
        System.out.println("-----------------------------\n");

        System.out.println("---Segitiga S2 Baru---");
        System.out.println("Alas S2     : " + S2.getAlas());
        System.out.println("Lebar S2    : " + S2.getTinggi());
        System.out.println("Pytagoras S2: " + S2.pytagoras());
        System.out.println("Luas S2     : " + S2.luas());
        System.out.println("-----------------------------\n");
```

11. Jika sudah, **Run** program kalian dengan cara mengKlik tombol Play pada toolbar menu di atas. Bisa juga menggunakan **F6**.

12. Silahkan cek value-value yang ada, apakah sesuai dengan apa yang kalian inputkan atau tuliskan atau tidak. Hasilnya kurang lebih akan seperti ini.

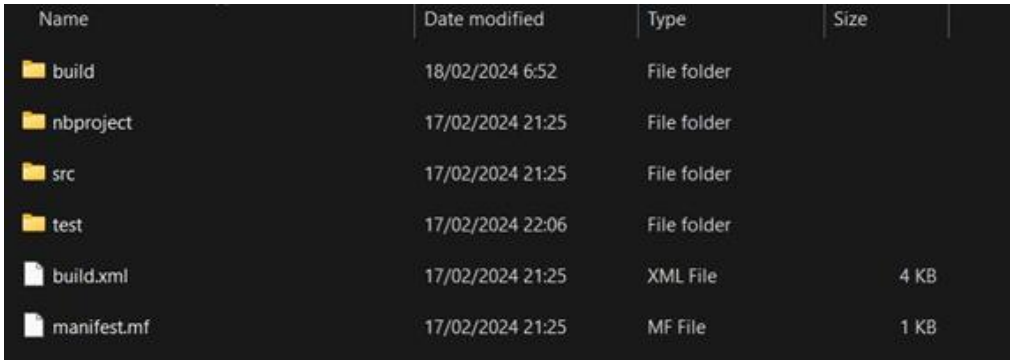
```
run:
---Segitiga S1 Lama---
Alas S1     : 10.2
Lebar S1    : 12.4
Pytagoras S1: 16.056151469141042
Luas S1     : 63.239999999999995
------------------------------

---Segitiga S2 Lama---
Alas S2     : 3.5
Lebar S2    : 4.2
Pytagoras S2: 5.4671747731346585
Luas S2     : 7.3500000000000005
------------------------------

--- Input Alas dan Tinggi BARU ---
Masukkan Nilai Alas yang baru :
3
Masukkan Nilai Tinggi yang baru :
4.0
---Segitiga S1 Baru---
Alas S1     : 10.2
Lebar S1    : 12.4
Pytagoras S1: 16.056151469141042
Luas S1     : 63.239999999999995
------------------------------

---Segitiga S2 Baru---
Alas S2     : 3.0
Lebar S2    : 4.0
Pytagoras S2: 5.0
Luas S2     : 6.0
------------------------------

BUILD SUCCESSFUL (total time: 1 minute 1 second)
```

Dari kode di atas mengapa yang berubah hanya segitiga S2 ?
Karena kita **hanya memanggil mutator/setter dari Objek S2 saja, sedangkan S1 tidak** lakukan, maka dari itu value dari persegi panjang S1 akan tetap seperti value di konstruktor default.

Jika kalian mempunyai waktu yang longgar dan ingin berlatih lagi kalian bebas untuk menambah bangun datar yang lainnya. tidak perlu membuat project baru, cukup menambahkan kelas baru saja.

Ketentuan Pengumpulan Guided

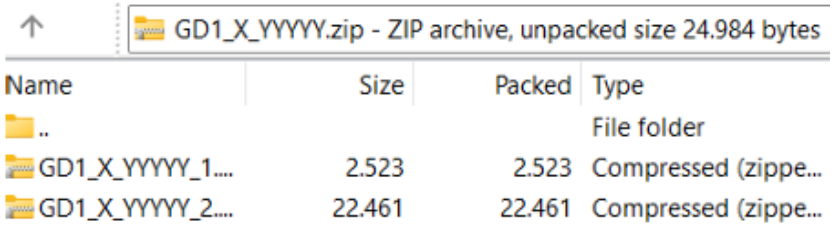
1. Ketika kalian sudah membuat project, maka didalam folder **GD1_X_YYYYY_2** kalian terdapat file yang dibuat oleh NetBeans, seperti pada contoh berikut ini:



2. Zip lah folder **GD1_X_YYYYY_2**

Ketentuan Pengumpulan Guided ke Situs Kuliah

GD1_X_YYYYY_1.zip dan **GD1_X_YYYYY_2.zip** dijadikan satu. Dizip lagi dengan nama **GD1_X_YYYYY**



Hint:
Silakan mempelajari penggunaan BufferedReader dengan tipe data lainnya (termasuk mempelajari penggunaan berbagai tipe data dalam sebuah kelas yang berbeda).

Selain itu, silakan membaca atau menonton film-film kartun, karena hal tersebut akan membantu kalian dalam mengerjakan UGD nantinya.