MODUL 3
“Exception, I/O, dan Operasi File”

Rekayasa Perangkat Lunak
(Software Engineering)
SMKN 2 TRENGGALEK

Oleh
SAFIRA MAYA SHOVIE, S.Pd

Exception, I/O, dan Operasi File

I. TUJUAN PEMBELAJARAN

a. Mengerti konsep exception, I/O, dan operasi file

b. Mampu mengimplementasikan konsep exception dan I/O dalam sebuah pemrograman sederhana untuk operasi file dengan tepat.

II. DASAR TEORI

a. Pendahuluan

Sering kali, ketika menjalankan sebuah program kita menemukan kesalahan. Kesalahan itu bisa berasal dari hardware, software, atau bahkan dalam algoritma atau kesalahan logika itu sendiri, jadi masing-masing kesalahan punya solusi yang beda-beda dan teknik penanganannya juga berbeda-beda untuk kesalahan yang satu dengan yang lain

Kesalahan itu terdiri dari :

1. Kesalahan Hardware dan Software

Hardware saja

Jika sebuah aplikasi berusaha membuka sebuah file yang tidak ada, misalnya mengirim karakter ke printer dalam kondisi Off atau ketika berkomunikasi dengan port serial yang tidak merespon.

Software saja

Ketika kode berusaha mengakses sebuah elemen yang di luar ikatan array atau berusaha menyimpan nilai yang melewati kapasitas format data. Kesalahan yang terkait sama software, harus dideteksi dengan kode

Hardware dan Software

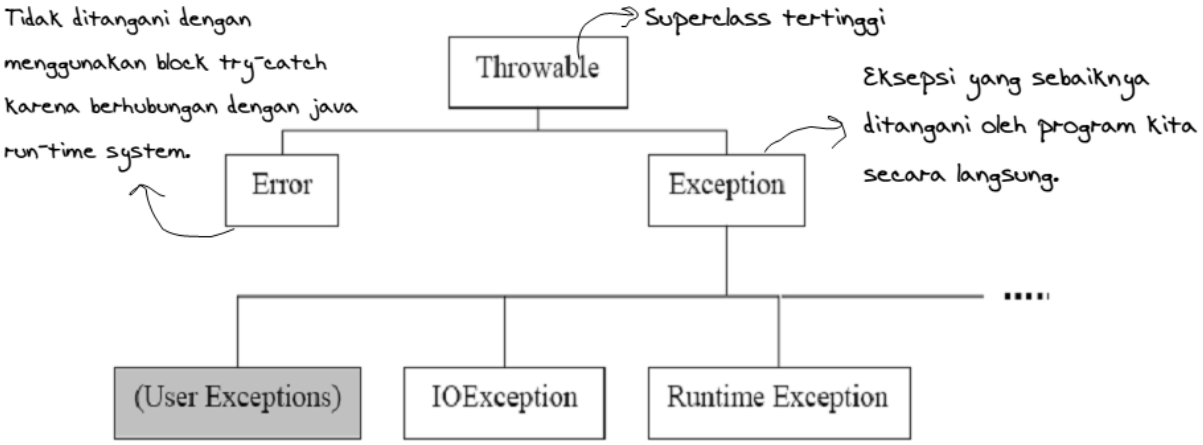
sebuah aplikasi mungkin memeriksa operan pembagi untuk memastikan bahwa sebuah pembagi dengan nol tidak dicoba. Bagaimanapun juga, jika sebuah pembagian oleh nol terjadi, perangkat-keras dalam sebagaian besar sistem komputer menghasilkan sebuah respon kesalahan.

2. Kesalahan Algoritma

Berhubungan dengan kesalahan atau keterbatasan intrinsik dari pemodelan dunia nyata yang dilakukan oleh komputer. Untuk mengatasi kesalahan pembulatan dan pemotongan pada berbagai algoritma muncullah disiplin ilmu bernama analisa numerik. Jenis kesalahan ini adalah yang paling sering diabaikan oleh programmer

b. **Exception**

Exception adalah kondisi abnormal / tidak wajar yang terjadi pada saat pengeksekusian suatu perintah. Dalam java ada lima keywords yang digunakan untuk menangani eksepsi ini yaitu : **try, catch, finally, throw**, dan **throws**.



Semua class exception terdapat dalam package java.lang.

Semua yang bertipe Runtime Exception dan turunannya tidak harus ditangani dalam program kita

Contoh program dibawah ini :

```
class RuntimeException
{
   public static void main (String [] args)
   {
      int[] arr = new int[1];
      System.out.println(arr[1]);
   }
}
```

Output :

```
Exception in thread "main" java.lang.
ArrayIndexOutOfBoundsException: 1 at RuntimeException .main
(RuntimeException.java:6)
```

Terjadi kesalahan ArrayIndexOutOfBoundsException karena array pada contoh diatas Cuma punya 1 record arr[0], tapi program menampilkan arr[1].

1. Pengunaan Try Catch

Untuk menangani eksepsi yang terjadi dalam program, Anda cukup meletakkan kode yang ingin anda inspeksi terhadap kemungkinan eksepsi di dalam blok try, diikuti dengan blok catch yang menentukan jenis eksepsi apa yang ingin Anda tangani.

Struktur penulisan:

```
try{
            ......//blok program
} catch (tipeException e) {
            ......//blok program
}
```

Latihan 1 :

TryCatch1.Java

```
public class TryCatch1 {
    public static void main (String[] args){
       try{
          int[] arr = new int[1];
          System.out.println(arr[1]);
          System.out.println("Baris ini tidak akan dieksekusi,
          karena statement baris diatas terjadi exception");
       }catch (ArrayIndexOutOfBoundsException e){
          System.out.println("Terjadi exception karena indeks di
          luar kapasitas array");
       }
       System.out.println("Setelah blok try catch");
    }
}
```

TryCatch2.Java

```
public class TryCatch2 {
    public static void main (String [] args){
       try{
          int x = args.length; //banyak argumen
          int y = 100/x;
          int[] arr = {10,11};
          y = arr[x];
          System.out.println("Tidak terjadi exception");
       }catch (ArithmeticException e){
          System.out.println("Terjadi exeption karena
          pembagian dengan nol");
       }catch (ArrayIndexOutOfBoundsException e){
          System.out.println("Terjadi exeption karena indeks
          di luar kapasitas array");
       }
       System.out.println("Setelah blok try catch");
    }
}
```

2. Pengunaan Throw

Mekanisme throw memungkinkan program untuk mengirim / melempar exception untuk kemudian program menyikapi (bereaksi) atas exception tersebut. Objek Eksepsi adalah semua objek yang merupakan turunan dari class Throwable. Salah satu contoh objek eksepsi adalah ArrayIndexOutOfBounds.

Cara penulisan:

```
Throw ObjekEksepsi;
```

Cara penulisan:

```
class NamaClass throws Exception{
            .........
}

atau fungsi / prosedur

int f1() throws Exception{
            ..........
}
```

Latihan 2 :

CobaThrow.Java

```
public class CobaThrow{
   public static void methodLain() {
      try{
         throw new ArrayIndexOutOfBoundsException(1);
      }catch(ArrayIndexOutOfBoundsException e) {
         System.out.println("Penanganan exception
         dalam method methodLain()");
         throw e;
   }
}
public static void main (String [] args){
   try{
      methodLain();
      }catch (ArrayIndexOutOfBoundsException e) {
         System.out.println("Penanganan exception
         dalam method main()");
      }
   }
}
```

3. Penanganan Dengan Finally

Penggunaan blok **try catch** terkadang membingungkan karena kita tidak dapat menentukan dengan pasti alur mana yang akan dieksekusi. Apalagi penggunaan throw yang mengakibatkan kode setelah throw tidak akan dieksekusi atau justru terjadi kesalahan pada blok catch, menyebabkan program akan berhenti. Untuk mengatasi problem ini, Java memperkenalkan keyword finally. Dimana semua kode yang ada dalam blok finally "pasti" akan dieksekusi apapun yang terjadi di dalam blok try catch.

Cara penulisan:

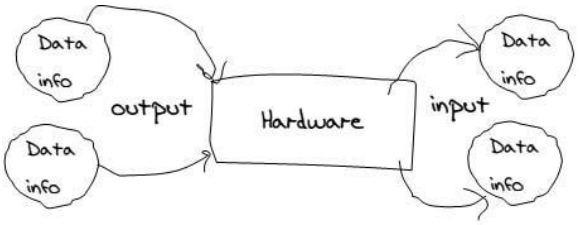
```
try{
            ......//blok program
} catch (tipeException1 e) {
            ......//blok program untuk
TipeException1
} catch (tipeException2 e) {
            ......//blok program untuk
TipeException2
} finally {
            ......//blok program
}
```

Latihan 3 :

DemoFinally.Java

```
public class DemoFinally{
  public static void main (String [] args){
    int x = 3;
    int [] arr = {10,11,12};
    try {
      System.out.println(arr[x]);
      System.out.println("Tidak terjadi exception");
    }catch(ArrayIndexOutOfBoundsException e) {
      System.out.println("Terjadi exception");
      System.out.println(arr[x-4]);
    }
    finally {
    System.out.println("Program Selesai");
    }
  }
}
```

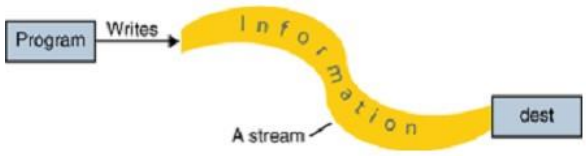
c. I/O (Input dan Output)



Input: mengambil informasi data dari hardware
Output: mengirim informasi data ke hardware

sebuah program membuka stream dari sebuah sumber informasi (source) seperti file, memory, socket. Kemudian membaca informasi secara sekuensial.

Stream adalah aliran proses informasi data yang direpresentasikan secara abstrak untuk untuk menulis/menghasilkan/output dan membaca/mendapatkan suatu informasi/input.
Semua streams punya sifat yang sama meskipun peralatan fisik yang berhubungan dengannya beda-beda (missal: keyboard, jaringan, dll.)

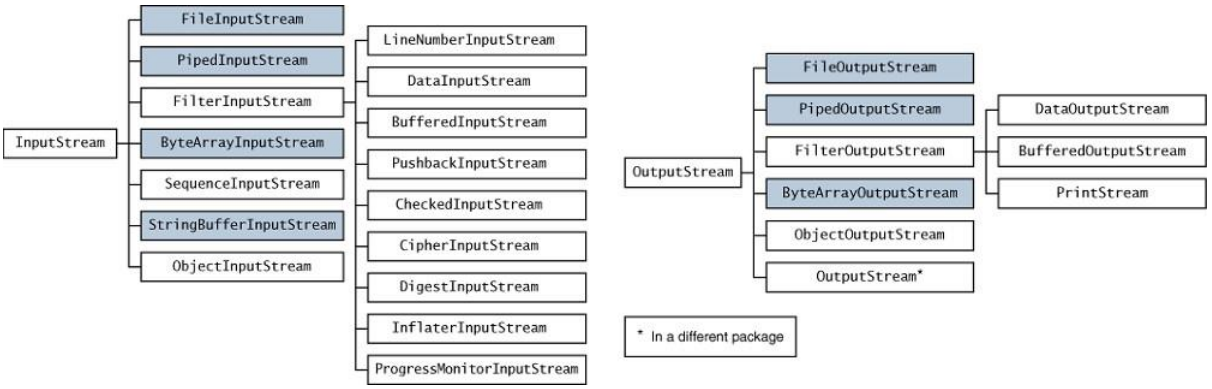


sebuah program dapat mengirim informasi dengan membuka stream ke sebuah tujuan informasi (destination). Kemudian menuliskan informasi ke tujuan secara sekuensial.

Paket yang menangani pembacaan dan penulisan info untuk pemrograman java © **java.io**. Ada 2 tipe, **Byte Stream** dan **Character Stream**

1. **Byte Stream**

   Byte Streams digunakan untuk operasi I/O yang menggunakan data biner (byte). Pada java byte stream mempunyai 2 buah superclass yaitu **InputStream** dan **OutputStream** yang merupakan class abstract.



Latihan 4 :

Demo Stream1.Java

```
import java.io.*;
public class DemoStream1
{
   public static void main(String[] args) {
      byte[] data = new byte[10];
      System.out.print("Masukkan data : ");
      try {
         System.in.read(data);
      } catch (IOException e) {
         System.out.print("Terjadi Exception");
      }
      System.out.print("Yang anda ketik : ");
      for (int i=0;i<data.length;i++) {
         System.out.print((char)data[i]);
         //(cahr) diatas disebut posting yakni untuk
         mengubah format menjadi char
      }
   }
}
```

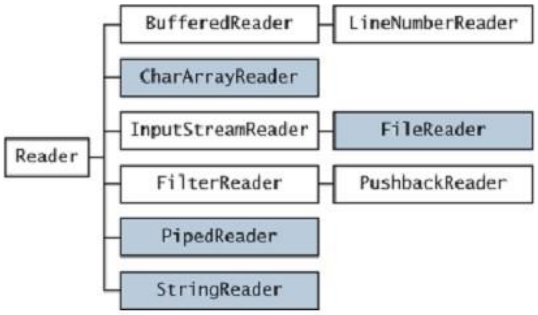
Demo Stream2.Java

```
import java.io.*;
public class DemoStream3
{
   public static void main(String[] args) {
      byte[] data = new byte[10];
      int panjang=0;
      System.out.print("Masukkan data : ");
      try {
         panjang=System.in.read(data);
      //Sistem.in.read mengembalikan panjang karakter yang
      //diinputkan (termasuk enter yang dianggap 2 karakater..)
         System.out.print("Yang anda ketik : ");
         System.out.write(data);
         System.out.println("Panjang Karakter : "+panjang);
         System.out.print("index ke-1 sebnyk 3 : ");
         System.out.write(data,1,3);
      } catch (IOException e) {
         System.out.print("Terjadi Exception");
      }
   }
}
/* write mencetak apapun tipe data yang ada, sedangkan print dan println mencetak data ke
dalam tipe string */
```

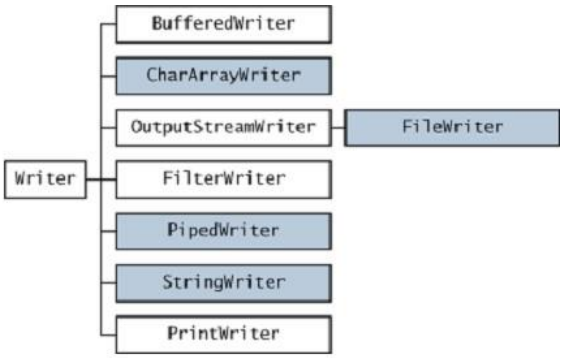
2. Character Stream

Apa itu character stream ?

Digunakan untuk menangani operasi I/O yang menggunakan character dan merupakan sebuah objek yang dapat membaca dan menuliskan **byte stream**. Jadi, character stream itu adalah sebuah byte stream yang diteruskan oleh class **Reader** dan **Writer** yang merupakan **Abstract class**.



Hierarki Kelas Reader



Hierarki Kelas Writer

Latihan 5 :

Demo Stream3.Java

```
import java.io.*;
public class DemoStream3 {
   public static void main(String[] args) throws IOException {
      char data;
      String str="";
      BufferedReader buff =
      new BufferedReader(new InputStreamReader(System.in));
      System.out.print("Ketik : ");
      data = (char) buff.read();
      while (data!='\r') {
      str+=data;
      data = (char) buff.read();
   }
   System.out.println("Yang diketik : "+str);
   System.out.println("Program Selesai");
   }
}
```

d. **I/O (Input dan Output)**

File digunakan sebagai media penyimpan. Untuk mengakses file kita harus menspesifikasikan dimana file yang akan kita akses, atau file baru yang akan disimpan. Dalam java kita dapat melakukan operasi file, yaitu membuat file baru atau menulis dan membaca file dengan character stream atau dengan byte stream. Untuk menciptakan sebuah file dengan mengakses class java.io.File dan menciptakan objek dari class tersebut, ini tanpa harus menangkap error io. Berbeda dengan menciptakan file yang langsung diakses oleh stream, file tersebut harus dapat menangkap error io ketika penciptaan objek class file.

- **java.io.File**

  terdapat 4 atribut, 4 konstruktor dan 39 method yang ada didalam class untuk menspesifikasikan file yang dibuat.

  File(),File(String path),File(String dir,String nm) ⟶ Konstruktor diatas adalah yang sering digunakan yaitu membuat objek file kemudian digunakan dengan pengesetan methodnya, atau menginstan langsung dengan nama file beserta pathnya.

  boolean createNewFile(), boolean delete(),boolean exists() ⟶ method-method diatas untuk mengeset dengan pengecekan, untuk createNewFile digunakan untuk menciptakan file kemudian mengembalikan nilai true jika file dibuat.

- **java.io.FileInputStream** ⟶ dengan byte stream

  terdapat 3 konstruktor dan 9 method yang ada. Digunakan untuk mengambil file yang telah dideskripsikan untuk dibaca dengan byte stream.

  FileInputStream(File of), FileInputStream(String nama) ⟶ Digunakan untuk mengambil file untuk dibaca secara byte stream,bisa memasukan deskripsi file yang telah ada dengan String, atau dengan file yang telah diinstan dengan jelas.

- **java.io.FileWriter** ⟶ dengan character stream

  terdapat 5 konstruktor dan tidak ada method yang dideskripsikan didalam class ini.

  FileWriter(File of), FileWriter(File of,boolean append) ⟶ Digunakan untuk penciptaan objek file yang akan diakses dengan character stream, dan untuk variabel append digunakan untuk apakah isi file akan dilanjutkan ke akhir dari isi file.

- **java.io.FileOutputStream** ⟶ dengan byte stream

  terdapat 5 konstruktor dan 7 method untuk membuat file yang akan diakses menggunakan byte stream.

  FileOutputStream(File of), FileOutputStream(File of,boolean append) digunakan untuk penciptaan objek file yang akan diakses dengan byte stream, dan untuk variabel append digunakan untuk apakah isi file akan dilanjutkan ke akhir dari isi file.

- **java.io.FileReader** ⟶ dengan character stream

  terdapat 3 konstruktor dan tidak ada method yang dideskripsikan didalam class ini.

  FileReader(File of), FileReader(String nama) ⟶ Digunakan untuk mengambil file untuk dibaca secara character stream,bisa memasukan deskripsi file yang telah ada dengan String, atau dengan file yang telah diinstan dengan jelas.

Latihan 6 :

Demo Stream4.Java

```
import java.io.*;
public class DemoStream4 {
  public static void main(String[] args) {
    byte data;
    String namaFile = "test.txt";
    FileOutputStream fout = null;
    try  {
      fout = new FileOutputStream(namaFile, true);
      //true artinya menambahkan kedalam file, tidak menimpa
      System.out.print("Ketik : ");
      data = (byte)System.in.read();
      while (data!=(byte)'\r') {
      fout.write(data);
      data = (byte)System.in.read();
      }
    }
    catch (FileNotFoundException e) {
      System.out.println("File "+namaFile+" tidak dapat
      dicreate");
    }
    catch (IOException e) {
      System.out.println("Terjadi Exception");
    }
    finally {
      if (fout!=null) {
      try {
      fout.close();
    }
    catch (IOException e) {
      System.out.println("Terjadi Exception");
    }
    }
    }
  }
}
```

e. **Object Serialization**

1. **Apa itu object serialization?**

   Suatu mekanisme yang dapat membuat sebuah objek dapat dikirimkan seperti mengirimkan data. Object serialization merupakan perluasan dari inti class java io yang digunakan untuk objek dan bisa digunakan untuk pengkodean (encoding) untuk objek dan membuat objek tersebut dapat diraih atau digunakan, dengan melalui bit-bit stream, kemudian dapat digunakan untuk penyusunan kembali objek tersebut dari bit-bit stream yang dikodekan, dan saling melengkapi. Serialisasi merupakan mekanisme yang ringan dan kuat untuk komunikasi dengan sockets atau RMI(Remote Method Invocation). Selain untuk komunikasi dengan sockets teknik ini dapat digunakan juga untuk menyimpan keadaan suatu status dari suatu objek ke dalam file, seperti yang sudah dijelaskan di pendahuluan. Jika kita ingin membuat sebuah objek yang dapat diserialisasikan, maka kita harus mengimplementasikan salah satu interface java.io.Serializable atau java.io.Externalizable pada class yang ingin dibuat objek yang dapat diserialisasikan. Untuk lebih jelas tentang serialisasi objek dibawah ini akan diberikan sedikit penjelasan yang terkait dengan objek serialisasi.

2. **Interface java.io.Serializable**

   Interface serializable harus di implementasikan jika ingin membuat objek yang dapat diserialisasi implementasi interface serializable tergolong sederhana karena tidak terdapat method yang harus didefinisikan untuk di override. Tujuan mengimplementasikan interface serializable adalah untuk memberitahukan kepada JVM (Java Virtual Machine), bahwa objek yang menerapkan serializabel merupakan objek yang dapat diserialisasikan.

   - **Class java.io.ObjectOutputStream**

     Class objectoutputstream adalah kelas yang digunakan untuk mengirimkan objek menjadi stream yang kemudian dapat dikirimkan ke file atau ke socket (jaringan). Class ObjectOutputStream mempunyai 2 constructor dan 31 method untuk versi jdk 1.5. Adapun method dan constructor yang sering digunakan adalah :

     ObjectOutputStream(OutputStream out) → Membuat ObjectOutputStream yang akan menuliskan ke spesifik OutputStream yang dikehendaki.

     void writeObject(Object obj) → menuliskan objek yang akan dikirimkan ke ObjectOutputStream

   - **Class java.io.ObjectInputStream**

     Class ObjectInputStream adalah kelas yang digunakan untuk mengambil objek dari stream yang dikirimkan melalui file atau socket (jaringan). Class ObjectInputStream mempunyai 2 Constructors, 31 methods untuk versi jdk 1.5. Method dan constructor yang sering digunakan adalah :

     ObjectInputStream(InputStream in) → Membuat ObjectInputStream yang akan mengambil spesifik stream dari InputStream yang dikehendaki.

     Object readObject() → Membaca objek yang telah diddefinisikan.

Latihan 7 :

```
import java.io.*;
public class BarangSer implements Serializable{
   private String nama;
   private int jumlah;

   public BarangSer (String nm, int jml){
      nama=nm;
      jumlah=jml;
   }

   public void tampil(){
      System.out.println("nama barang: "+nama);
      System.out.println("jumlah barang: "+jumlah);
   }
   public void simpanObject(BarangSer ob){
      try{
         FileOutputStream fos= new FileOutputStream("dtBrg.txt");
         ObjectOutputStream oos = new ObjectOutputStream(fos);
         oos.writeObject(ob);
         oos.flush();
      }catch(IOException ioe){
         System.err.println("error"+ioe);
   }
   }
   public void bacaObject(BarangSer obb){
      try{
         FileInputStream fis= new FileInputStream("dtBrg.txt");
         ObjectInputStream ois = new ObjectInputStream(fis);
         while( (obb=(BarangSer)ois.readObject())!=null)
         obb.tampil();
      }catch(IOException ioe){
         System.exit(1);
      }catch(Exception e){
         System.exit(1);
      }
   }

public static void main(String[] args){
   BarangSer a1 = new BarangSer("Baju",5);
   a1.simpanObject(a1);
   a1.bacaObject(a1);
   }
}
```

```
import java.io.*;
public class BarangEx implements Externalizable{
   private String nama;
   private int jumlah;

   public BarangEx(){            //konstruktor 1
   }

   public BarangEx(String nm, int jml){     //konstruktor2
      nama=nm;
      jumlah=jml;
   }
   public void writeExternal(ObjectOutput out) throws IOException{
      out.writeObject(nama); //string adalah tipe data referensi
      out.writeInt(jumlah);
   }

   public void readExternal(ObjectInput in) throws IOException,
   ClassNotFoundException{
      this.nama = (String) in.readObject();
      this.jumlah= in.readInt();
   }
   public String toString(){
   return "data barang: "+nama+"\n"+"jumlah barang: "+jumlah;
   }
   public static void simpanObjek(BarangEx brg) throws IOException{
      FileOutputStream fos = new FileOutputStream("dtEx.txt");
      ObjectOutputStream oos=new ObjectOutputStream(fos);
      oos.writeObject(brg);
      oos.flush();
   }
   public static BarangEx bacaObjek() throws
   ClassNotFoundException,IOException{
      FileInputStream fis= new FileInputStream("dtEx.txt");
      ObjectInputStream ois = new ObjectInputStream(fis);
      return(BarangEx)ois.readObject();
   }
   public static void main(String[] args) throws
   ClassNotFoundException,IOException{
      BarangEx awal = new BarangEx("sepatu",2);
      simpanObjek(awal);
      System.out.println(bacaObjek());
   }
}
```

Catatan :

1. Silahkan mengerjakan semua latihan yang terdapat diatas, bisa didiskusikan dengan teman kalian. Tetapi untuk pengerjaan tugasnya tetap individu.
2. Materi diatas hanya sedikit rangkuman dari berbagai fungsi kode program yang sudah saya buat. Silahkan untuk pemantaban pemahaman lebih dalam, kalian cari referensi contoh program yang lain dengan menggunakan beberapa fungsi dari kode program diatas.
3. Deadline pengumpulan tugas, 4 hari setelah materi dibagikan. Jika telat dalam pengumpulan, tugas tidak diterima, terkecuali bila ada kendala, misal : laptop rusak dsb, maka setelah laptop kembali normal tugas bisa menyusul.
4. Tugas dikumpulkan melalui link berikut : https://drive.google.com/drive/folders/1PPC2f0s4kFR42O6KKSHvYlif_eLJml4C?usp=sharing , dengan catatan yang dikumpulkan adalah project pada netbeans yang sudah dikerjakan, dan laporan yang berisi screen shoot syntax program dan output program serta penjelasan program.
5. Jangan lupa format folder tugas dan laporan yaitu **No.Absen_Nama Lengkap_Kelas**.

SELAMAT MENGERJAKAN ^_^