

Week 8 - Abstraction

- Abstraction merupakan konsep menyembunyikan detail serta kompleksitas dari suatu implementasi ke user.
- Abstract Class merupakan
 - salah satu bentuk wujud implementasi abstraction.
 - salah satu tipe class dlm OOP yang dapat memiliki 1/lbh abstract method.

◦ Concrete Class	Abstract Class
<p>Mirip class-class biasa</p> <p>Dapat inisialisasi object dgn keyword → new ClassName(....)</p> <p>Tiap fungsi hrs secara langsung diimplementasi (Concrete methods)</p>	<p>—</p> <p>Tidak dapat inisialisasi object</p> <p>Tiap fungsi tidak hrs secara langsung diimplementasi (Concrete methods / Abstract methods)</p>

```
◦ public abstract class ClassName{  
    access_modifier data_type_1 attribute_11, ..., attribute_1n;  
    ...  
    access_modifier data_type_2 attribute_21, ..., attribute_2n;  
  
    public ClassName(args...){ ... }  
  
    // abstract method  
    public abstract returned_type method_name_1(args...);  
    ...  
    // concrete method  
    public returned_type method_name_1(args...){...}  
    ...  
}
```

- Concrete & Abstract class dpt memiliki constructors, attributes, atau concrete method
- Pada abstract class, dpt tambh keyword abstract pada sebuah method utk nyatakan method belum memiliki detail implementasi

- Abstract juga bs extends
- Abstract class dpt digunakan sbg template / format (Baik properti / fungsi) utk class-class turunannya.
- Method yg bersifat abstract pd class turunan concrete akan error
- Abstract class dapat menampung objek-objek yang ada dlm class turunannya
- Class diagram abstract → Italic
- Abstract class = Restricted class yg gbs buat objek (Akses hrs dr inherited dr class lain)
- Abstract method = Hny bs dipakai di abstract class & gada body. (Body disediakan oleh subclass diherited)

```

// Abstract class pertama
public abstract class Hewan {
    public abstract void bersuara();
}

// Abstract class kedua yang menurunkan Hewan
public abstract class Mamalia extends Hewan {
    // Tidak mengimplementasikan bersuara()
}

// Concrete class yang menurunkan Mamalia
public class Anjing extends Mamalia {
    @Override
    public void bersuara() {
        System.out.println("Menggonggong");
    }
}

```

Week 9 - Multiple Inheritance

- **Multiple inheritance** merupakan kondisi dimana sebuah kelas dapat diwarisi oleh lebih dari 1 kelas baik secara langsung / tidak langsung
- Java tidak izinkan class (Abstract / Concrete) untuk diwarisi oleh lebih dari 1 class lainnya
 - ↳ If want then hrs scr tdk lgsg dan bertahap proses pewarisanannya.
- Java ksh multiple inheritance scr lgsg melalui interface

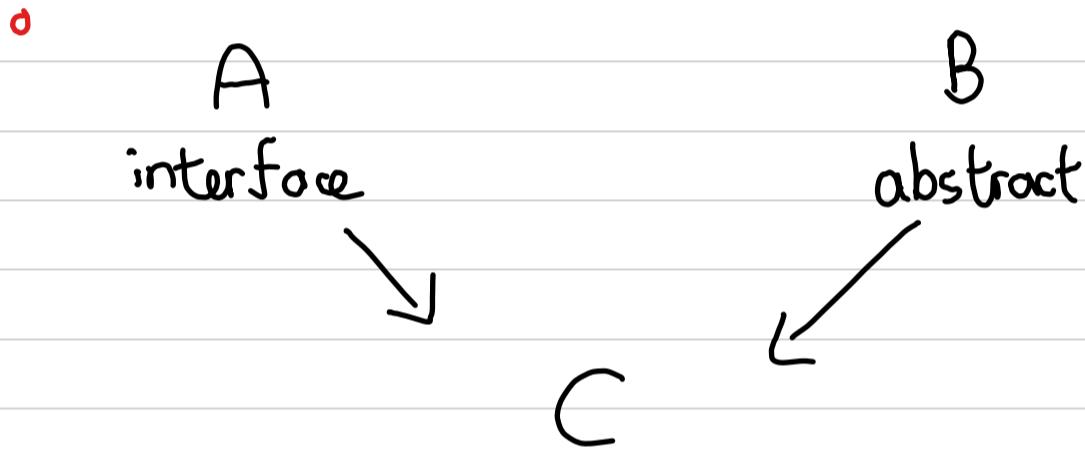
```
○ public class A{  
    protected String a;  
    public void a() { ... }  
}  
  
public class B extends A{  
    protected String b;  
    protected void b(){ ... }  
}
```

```
public class C extends B{  
    protected String c;  
    public void c() { ... }  
}  
  
public class D extends C{  
    protected String d;  
    protected void d(){ ... }  
}
```

↳ D have all properti & method yang bersifat protected dan public pada class A, B, C

- Interface → Implementasi fitur abstraction pd OOP
→ Sbg Abstract class yang lebih terbatas
- Interface dpt digunakan untuk sediain template (Abstract method) untuk class tanpa liat hierarki dari class & implementasi hal yang bersifat umum.
- Interface → Variabel konstan + No concrete method
Abstract → Variabel konstan / nonkonstan + Concrete method

- o Use interface jika ingin sediakan template lbh general saat diwarisi ke class lain (Dpt implementasi ke all class tanpa liat detail properti yang dimiliki class)
- o Use abstract jika ingin sediakan template yg lbh spesifik
- o Interface gblh extends, hrs implements
- o Default interface → Public, static, final, abstract
- o Interface diwariskan ke interface pakai extends



class C extends B implements A {
}

o public interface Child extends A,B ✓

public interface Child implements A,B → Error

public Child implements A,B ✓

Week 10 - Final Keyword and Java Type Casting

- Final = Keyword khusus pada Java yang memiliki bermacam kegunaan tergantung keyword digunakan
 - Final keyword :
 - 1. Variables
 - 2. Classes
 - 3. Methods
 - Immutable
- Final class tdk dpt inherit
- Method juga tidak bisa override kalau final
- Reference Variables = Inisialisasi sebuah class object pada sebuah variabel, variabel tsb hny sbg penunjuk / reference dari class object yg dihasilkan
- Casting = Utk batasi sekumpulan properti / fungsi
- Casting < Upcasting
Downcasting
- Upcasting = Objek yg ditampung dalam reference bertipe child menjadi parent.
 - Member → VIP

VIP vipmember = new VIP()
Member memberref = vipmember
Member memberref = (Member) vipmember

} Scrimplist

◦ Downcasting = Objek yg dikampung dlm reference bertipe parent menjadi child.



Member → VIP

Member m = new VIP
VIP vm = (VIP) m

} Mungkin aj bs error
(Runtime exception)

↓
Atasi pakai instanceof

Week 11 - Exception Handling

- **Exception Handling** = Salah satu mekanisme / fitur pada Java yg buat alur eksekusi program tetap berjalan meskipun terdapat kesalahan (Runtime errors) pada program.
- **Exception** = Peristiwa / Kejadian tak terduga yg terjadi saat program sedang dieksekusi dan menghentikan alur eksekusi dari program
- **Error** = Kondisi series yang menghentikan program, tetapi tidak harus ditangani oleh aplikasi.
 - ↓
Kalau exception dpt dianggap kondisi yang juga menghentikan program tetapi hrsnya dpt diotasi ditangani
- Exception handling by using try, catch, finally (Opsional)
 - ↳ **Try** = Meletakkan potongan kode yg berpotensi mengembalikan exception
 - ↳ **Catch (Exception ex)** = Meletakkan potongan kode yg akan dijalankan ketika sebuah exception dengan tipe Exception terjadi
 - ↳ **Finally** = Put after try & catch selesai dijalankan
- **Exception** <
 - Checked exceptions (Ada error hrs debug)
 - Unchecked exceptions (Ada error gpp)

- Checked exceptions = Exceptions yg akan diperiksa saat program dicompile
 - ↳ FileNotFoundException = Ga ktmu file berdsrkan path yang dispesifikasikan
 - ↳ ClassNotFoundException = Berusaha mengembalikan objek berdsrkan sebuah path dimana source code terkait sebuah class berada
 - ↳ SQLException = Berusaha akses suatu sistem database
- Unchecked exceptions = Exceptions yang tidak akan diperiksa pada saat program dicompile
 - ↳ NullPointerException = Saat berusaha akses properti / method berdsrkan sebuah reference variabel yg menunjuk ke nilai null
 - ↳ ClassCastException = Saat berusaha casting pd objek yg talk sesuai dgn hierarki milik classnya
 - ↳ ArrayIndexOutOfBoundsException = Saat berusaha akses elemen dengan indeks yg gd di array
- Custom Exception = By using proses inheritance
- Untuk kembalikan sebuah exception, dpt menggunakan throw.

Week 12 - File Handling

- File Handling = Create, Read, Update, Delete file
- Create file :
 1. File file = new File ("asulah.txt")
 2. file. createNewFile()

↓
Try catch pakai IOException
- Read file : file. setReadable (true) → Bs bc
file. setWriteable (false) → Gbs tulis
Scanner myReader = new Scanner (file)
↑ tuk read file
- Write file : Menggunakan objek FileWriter
 1. FileWriter writer = new FileWriter ("ya.txt")
 2. writer. write (...)
 3. writer. close ()
- Append file : FileWriter writer = new FileWriter ("ya.txt", true)
- Delete file : file. delete ()

```
Scanner myReader = new Scanner(myObj);
while (myReader.hasNextLine()) {
    String data = myReader.nextLine();
    System.out.println(data);
}
```

Week 13 – Software Design Pattern

- Design Pattern = Suatu solusi umum utk atasi mslh dlm pengembangan sebuah software
- Design Pattern  Creational Design Pattern Behavioural
Behavioural Design Pattern
Structural Design Pattern
- Creational Design Pattern = Berhubungan dgn slrh hal yang terkait instansiasi objek
 - ↳ Abstract Factory
 - ↳ Builder
 - ↳ Factory Method
 - ↳ Object Pool
 - ↳ Prototype
 - ↳ Singleton

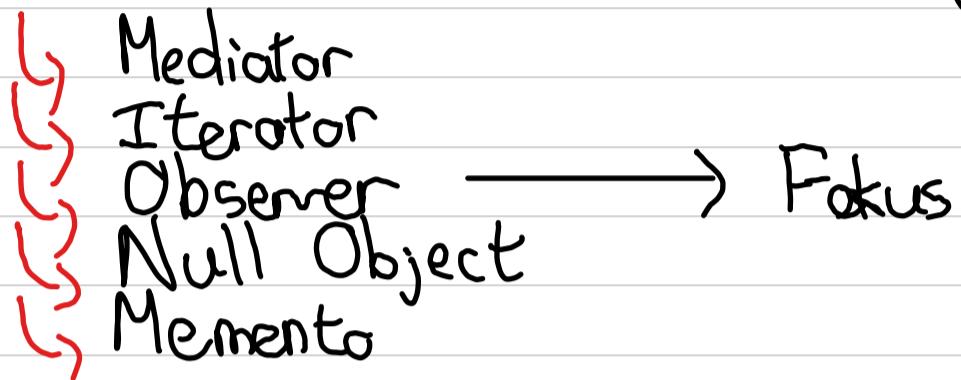
→ Fokus
- Object Pool = Mekanisme yg memanfaatkan sekumpulan objek yg sblnnya telah diinstansiasi dan tempatkan ke dalam wadah
 - ↳ Ga perlu alokasi memori & hapus memori
 - ↳ Digunakan pd software yg membutuhkan performa
 - Java, C# ada automatic garbage collector jd ga hrs manual

↓

Bsr dlm games & 3D simulation, then menyebabkan frame spike yg butuh sumber daya komputasi dr CPU besar
- Structural Design Pattern = Komposisi atribut/ method dr object
 - ↳ Adapter
 - ↳ Bridge
 - ↳ Composite
 - ↳ Proxy

→ Fokus

- Proxy = Sediain pengganti / placeholder utk sebuah objek yg hrs dipatuhi skmplt class utk akses
- Behavioural Design Pattern = Bagaimana objek-objek instansiasi sebuah class berinteraksi



- Observer = Definisikan 1to n relationship antara objek dari suatu class dgn objek lain
 - ↳ Saat state objek asal berubah, objek lainnya needs informasi saat interaksi dgn objek asal diinfokan