Rangkuman Materi untuk Sertifikasi Programmer

A. Software Process
   a. Menurut Pressman (2001), aktivitas-aktivitas yang dilakukan pada rekayasa perangkat lunak dapat dikategorikan menjadi 3 bagian besar yaitu:
      1. Definition Phase, pada fase ini software engineering berfokus pada apa/what. Informasi seperti apa yang akan diolah, fungsi dan kinerja seperti apa yang diinginkan, perilaku sistem seperti apa yang diharapkan, apa interfaces yang akan dibuat, hambatan apa saja yang akan dihadapi pada saat desain, dan kriteria apa saja yang harus dipenuhi agar sistem/aplikasi tersebut dapat dikatakan berhasil. Persyaratan utama dari sistem dan aplikasi ditentukan pada fase ini. Walaupun metode yang digunakan berbeda-beda, bergantung pada paradigma software engineering apa yang dipilih, namun terdapat 3 aktivitas utama yang selalu dilakukan, yaitu: system/information engineering, software project planning dan requirement analysis.
      2. Development Phase, pada fase ini software engineering berfokus pada “bagaimana/how”. Fase ini mencoba mendefiniskan bagaimana struktur data, bagaimana fungsi akan diimplementasikan ke dalam arsitektur perangkat lunak, bagaimana desain ake dalam bahasa akan diterjemahkan diterjemahkan ke dalam bahasa pemrograman, dan bagaimana testing akan dilaksanakan.Walaupun langkah-langkah yang akan dijalankan berbeda-beda, tetapi ada 3 aktivitas utama yang harus selalu dilakukan, yaitu: software design, code generation, dan software testing.
      3. Support Phase, pada fase ini software engineering berfokus pada “perubahan/change” yang dihubungkan dengan penanganan kesalahan, adaptasi yang diperlukan seiring dengan perubahan lingkungan perangkat lunak, dan perubahan yang dilakukan untuk menyempurnakan aplikasi sesuai dengan keinginan customer. Fase ini menjalankan kembali apa yang telah dilakukan pada 2 fase sebelumnya namun dalam konteks yang berbeda. Karena pada fase ini perangkat lunak sudah “jadi” tetapi perlu disempurnakan. 4 perubahan yang sering terjadi pada perangkat lunak yang dilakukan pada fase ini adalah: correction/pembetulan kesalahan, adaptation/adaptasi, enhancement/penyempurnaan, prevention/pencegahan.
   b. Menurut David A. Gustafson(2002), aktivitas-aktivitas tersebut dikelompokkan menjadi 8 kelompok utama, yaitu:
      1. **Feasibility**, menentukan apakah aplikasi atau perangkat lunak yang akan dibuat dapat dibuat atau tidak.Market analysis, menentukan apakah terdapat potensi pasar untuk aplikasi yang akan dibuat tersebut. Dokumen yang dihasilkan: Statement of work adalah dokumen yang deskripsi pendahuluan mengenai fungsi yang diinginkan, biasanya dibuat oleh user.
      2. **Requirement**, menentukan apa persyaratan dan fungsi yang akan dimiliki perangkat lunak yang akan dibuat. Requirement elicitation, mengumpulkan persyaratan dari pengguna. Domain analysis, menentukan ruang lingkup pekerjaan yang biasanya harus diselesaikan untuk di dalam permasalahan tersebut. Dokumen yang dihasilkan:
         a. Software requirement specification adalah dokumen yang menggambarkan apa yang bisa dilakukan oleh aplikasi ketika selesai diibuat
         b. Object model adalah dokumen yang menunjukkan object/class

    c. Use case scenarios adalah dokumen yang menunjukkan behaviors dari aplikasi ditinjau dari sudut pandang pengguna.
3. **Project planning**, menentukan cara mengembangkan aplikasi tersebut. Cost analysis, menentukan perkiraan biaya yang diperlukan. Scheduling, membuat penjadwalan untuk pengembangan aplikasi. Software quality assurance, menentukan aktivitas-aktivitas yang diperlukan untuk memastikan kualitas dari aplikasi tersebut. Work-breakdown structure, menentukan deskripsi pekerjaan yang diperlukan untuk mengembangkan aplikasi. Dokumen yang dihasilkan:
    a. Project schedule adalah dokumen yang menggambarkan urutan langkah yang akan dikerjakan dan perkiraan waktu pelaksanaan.
    b. Software quality assurance plan adalah adalah dokumen yang menggambarkan aktivitas apa yang akan dilakukan untuk memastikan kualitas dari aplikasi yang dibuat.
4. **Design**, menentukan bagaimana aplikasi tersebut harus memenuhi fungsi yang diinginkan. Architectural design, menggambarkan bagaimana struktur dari aplikasi. Interface design, menentukan interface antar bagian-bagian dari aplikasi. Detailed design, menentukan algoritma dari masing-masing bagian aplikasi. Dokumen yang dihasilkan:
    a. Software design, menggambarkan struktur dari aplikasi
    b. Architectural design, menggambarkan desain aplikasi secara makro berikut dengan hubungan antar bagian/modul (high level structure)
    c. Detailed design, menggambarkan desain aplikasi secara mikro sebagai modul atau objects (low level structure)
5. **Implementation**, membuat aplikasi. Dokumen yang dihasilkan: *Source Code*
6. **Testing**, menjalankan aplikasi yang telah dibuat untuk memastikan apakah aplikasi yang dibuat telah sesuai dengan persyaratan yang diinginkan dan berjalan dengan benar. Unit testing, testing yang dilakukan oleh developer. Integration testing, testing yang dilakukan pada saat penyatuan bagian-bagian dari aplikasi. System testing, testing perangkat lunak pada lingkungan yang sama dengan lingkungan dimana aplikasi tersebut digunakan. Alpha testing, testing yang dilakukan oleh user pada lingkungan developer. Beta testing, testing yang dilakukan oleh user pada lingkungan user. Acceptance testing, testing yang dilakukan untuk memuaskan keinginan user. Regression testing, menyimpan hasil testing yang dilakukan pada versi aplikasi sebelumnya untuk memastikan bahwa versi terbaru juga memiliki kemampuan yang sama dengan versi sebelumnya. Dokumen yang dihasilkan:
    a. Software test plan adalah dokumen yang menggambarkan bagaimana testing akan dilakukan pada aplikasi.
    b. Acceptance tests adalah testing yang dilakukan oleh customer untuk menentukan tingkat kepuasan pengguna berdasarkan standar yang telah ditetapkan.
    c. Test Report, dokumen yang menggambarkan apa test yang telah dilaksanakan dan bagaimana perilaku aplikasi terhadap tes tersebut.
7. **Delivery**, memberikan aplikasi yang efektif kepada customer. Installation, membuat aplikasi tersedia pada lokasi customer. Training, memberikan pelatihan kepada user agar dapat menggunakan aplikasi tersebut. Help desk, menjawab sema pertanya user mengenai aplikasi. Dokumen yang dihasilkan:

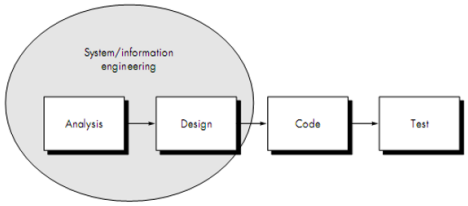
a. User manual adalah dokumen yang menjelaskan bagaimana menggunakan aplikasi

8. **Maintenance**, pengubahan dan perbaikan aplikasi untuk memastikan aplikasi tetap dapat digunakan. Dokumen yang dihasilkan:
    a. Defect Report adalah dokumen yang menjelaskan mengenai keluhan terhadap aplikasi yang dirasakan oleh customer, biasanya berupa laporan terhadap kesalahan yang dilakukan oleh aplikasi.

B. Siklus Hidup Perangkat Lunak adalah sekelompok aktivitas yang dilakukan secara berurutan dalam rangka mengembangkan sebuah perangkat lunak yang berulang terus menerus sehingga membentuk suatu siklus. Terdapat beberapa jenis siklus hidup yang dapat dipilih untuk digunakan oleh para software engineer yang bisa disesuaikan dengan perangkat lunak yang akan dikembangkan.

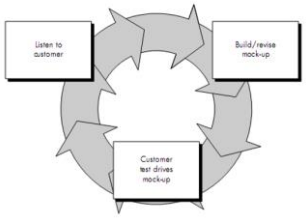
a. Linear Sequential Model

Biasanya disebut juga classic life cycle atau model waterfall adalah pemodelan yang menggunakan pendekatan sistematik dan berurutan untuk mengembangkan perangkat lunak yang diawali dengan level sistem dan dilanjutkan dengan analisa, desain, coding, testing dan support.



b. Prototyping Model

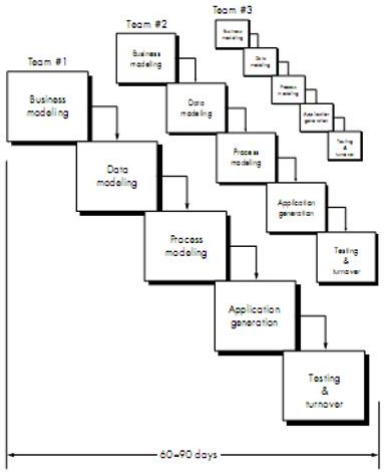
Pemodelan ini dimulai dengan melakukan requirement gathering. Developer dan customer bertemu dan membahas mengenai tujuan utama dari perangkat lunak yang akan dibuat, dan mencoba mengidentifikasikan semua requirement dan juga menentukan fungsi utama apa saja yang harus ada di dalam perangkat lunak tersebut. Setelah itu, developer akan mencoba membuat protipe/contoh dari perangkat lunak yang akan dibuat berdasarkan requirement yang telah disepakati sebelumnya. Prototipe ini, hanya berfokus pada aspek-aspek yang dapat dilihat oleh user seperti pendekatan input dan format outputnya. Kemudian customer akan mengevaluasi prototipe tersebut. Segala kesalahan dan kekurangan dari prototipe kemudian akan direvisi dengan perbaikan pada prototipe tersebut. Proses ini akan terus berulang hingga customer menyatakan persetujuannya. Prototipe akan dianggap sebagai versi pertama dari perangkat lunak dan idealnya tidak akan digunakan lagi (Brooks, 1975).



c. RAD Model

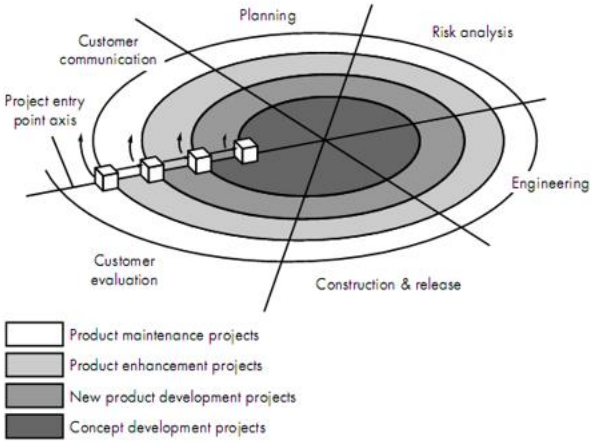
Rapid application development (RAD) adalah model pengembangan perangkat lunak bertingkat (incremental) yang menekankan pada pendeknya waktu pengembangan perangkat lunak. Pemodelan ini adalah adaptasi dari linear sequential model tetapi dengan waktu pengembang

yang lebih cepat. Karena pemodelan RAD menggunakan pendekatan komponen. Pengembangan aplikasi dengan menggunakan model ini dapat diselesaikan dalam waktu amat singkat sekitar 60 sampai 90 hari jika requirement yang ada benar-benar dipahami (Martin, 1991).



d. Spiral Model

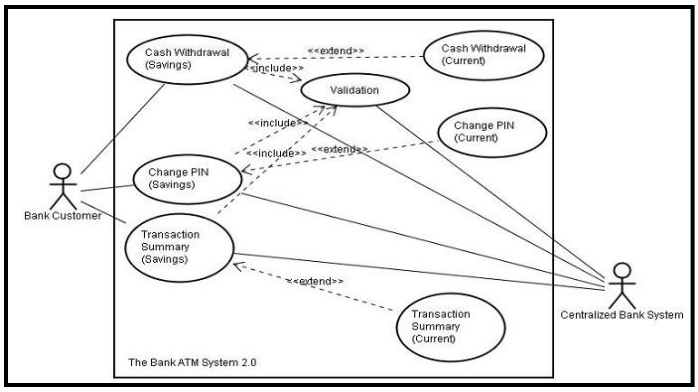
Spiral model adalah evolutionary software process model yang memasangkan prototype model dengan aspek sistematik dari linear sequential model (Boehm, 1995). Sebuah spiral model dibagi menjadi beberapa aktivitas yang disebut sebagai task region. Biasanya terdapat 3-6 task region dalam sebuah pengembangan perangkat lunak.



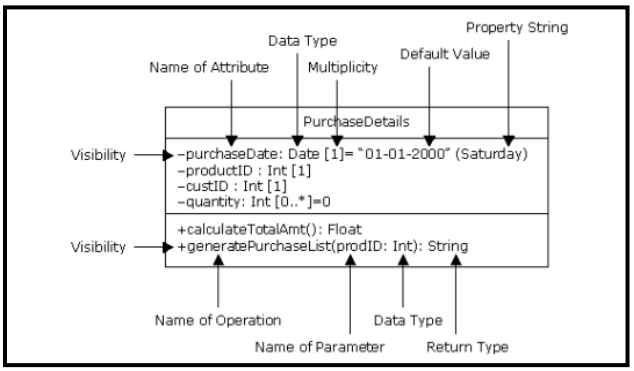
C. Diagram

a. UML (unified modelling language) memiliki beberapa jenis diagram dengan tujuan pembuatan yang berbeda-beda.

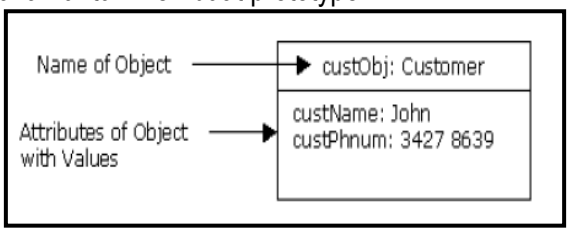
1. Use Case Diagram, diagram ini menggambarkan kumpulan use case, aktor, dan hubungan mereka. Use case adalah hubungan antara fungsionalitas sistem dengan aktor internal/eksternal dari sistem.



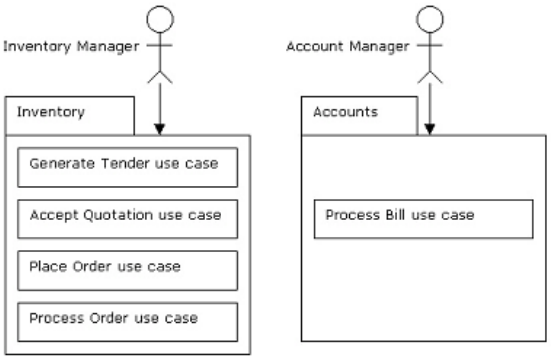
2. Class Diagram, diagram ini terdiri dari class, interface, association, dan collaboration. Diagram ini menggambarkan objek - objek yang ada di sistem.



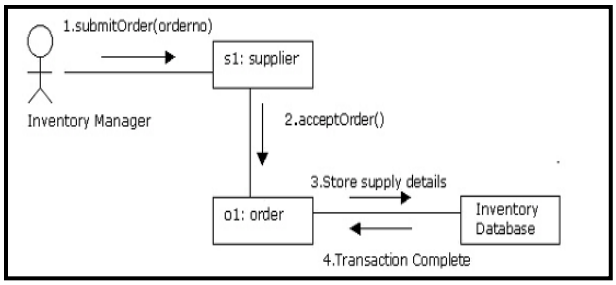
3. Object Diagram, agram ini menggambarkan hasil instansi dari class diagram. Diagram ini digunakan untuk membuat prototype.



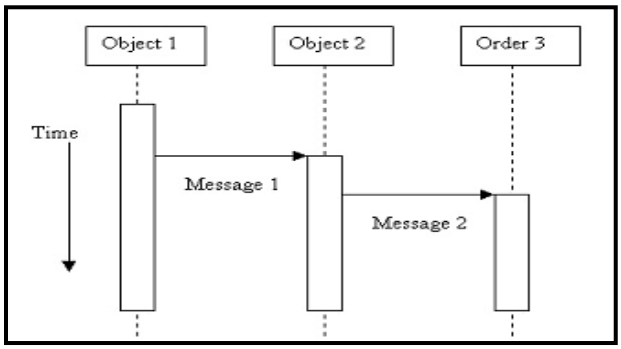
4. Package Diagram, diagram ini menggabungkan beberapa kelas sejenis dalam class diagram ke dalam package-package yang sama.



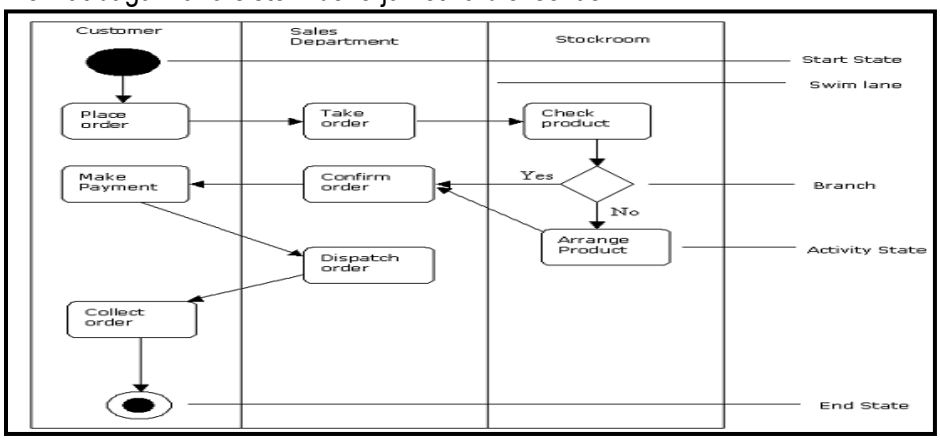
5. Communication Diagram, diagram ini merupakan bentuk lain dari sequence diagram. Diagram ini menggambarkan struktur organisasi dari sistem dengan pesan yang diterima dan dikirim.



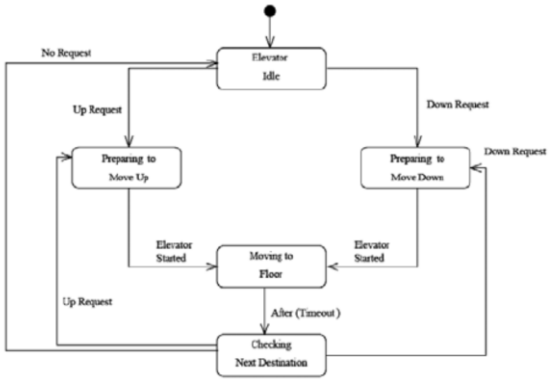
6. Sequence Diagram, diagram ini menggambarkan interaksi yang menjelaskan bagaimana pesan mengalir dari objek ke objek lainnya.



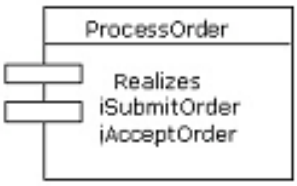
7. Activity Diagram, menggambarkan aliran kontrol sistem. Diagram ini digunakan untuk melihat bagaimana sistem bekerja ketika dieksekusi.



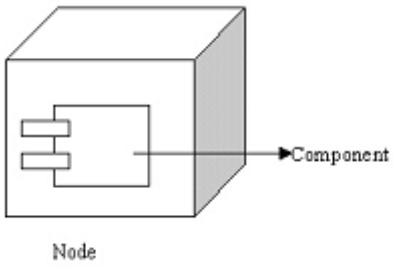
8. Statechart Diagram, diagram ini menggambarkan bagaimana sistem dapat bereaksi terhadap suatu kejadian dari dalam atau luar. Kejadian (event) ini bertanggung jawab terhadap perubahan keadaan sistem.



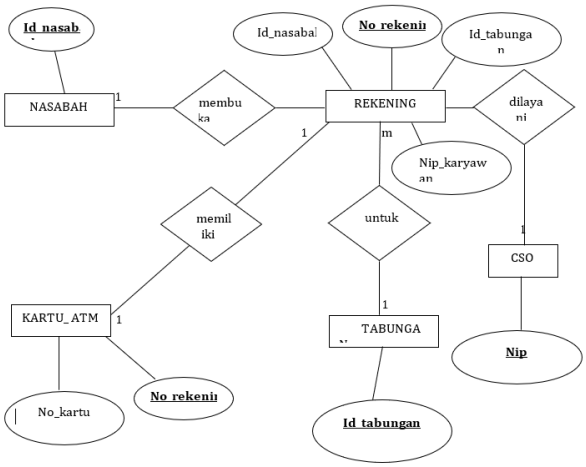
9. Component Diagram, diagram ini menggambarkan kumpulan komponen dan hubungan antar komponen. Komponen terdiri dari class, interface, atau collaboration



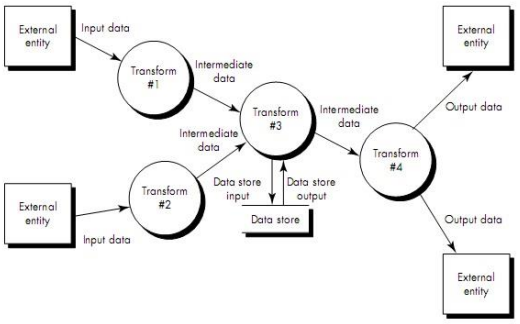
10. Deployment Diagram, diagram ini menggambarkan kumpulan node dan hubungan antar node. Node adalah entitas fisik dimana komponen di-deploy. Entitas fisik ini dapat berupa server atau perangkat keras lainnya.



b. ERD, adalah diagram yang menggambarkan Entity Relationship Model.



c. DFD (Data Flow Diagram) adalah diagram yang menggambarkan bagaimana data diubah ketika data tersebut mengalir di dalam sistem dan untuk menggambarkan fungsi dan atau sub fungsi yang merubah data tersebut



D. Component, adalah paket perangkat lunak, web service, atau sebuah modul yang menggabungkan sekelompok fungsi atau data yang terhubung satu dengan yang lainnya. Contoh component adalah: JavaBean, COM, COM+, DCOM, dan Web Servide.
   a. JavaBean adalah teknologi java yang merupakan software component dan dapat digunakan kembali Sebuah JavaBean menggabungkan banyak object menjadi hanya sebuah object yang dapat diakses melalui lokasi lain.
   b. COM, adalah cara mengimplementasikan sebuah object yang independent terhadap semua bahas pemrograman yang bahkan dapat diakses dengan menggunakan lingkungan yang berbeda dengan lingkungan dimana COM diciptakan. COM adalah komponen yang dibuat dengan menggunakan .Net Framework.
   c. COM+, adalah generasi terbaru dari COM yang mengakomodir adanya distributed transaction, resources pooling, disconnected application, event publication and subscription, manajemen memori dan prosesor yang baik.
   d. DCOM, adalah komponen yang berfokus pada distributed environment.
   e. Web Service, adalah perangkat lunak apapun yangtersedia melalui internet yang menggunakan standard messaging system dari XML

E. Object Oriented Programming
   a. Pendekatan berorientasi objek menggunakan suatu teknik yang disebut sebagai *object modelling*.
   b. *Object modeling* menggunakan metodologi dan diagram yang berbeda dengan *data modeling* dan *process modeling*.
   c. Object adalah entitas yang memiliki atribut, karakter (*behavior*), dan kondisi (*state*).
   d. Attributes merepresentasikan karakteristik dari suatu object.
   e. Method merupakan suatu operasi berupa fungsi-fungsi yang dapat dikerjakan oleh suatu object.
   f. Encapsulation (enkapsulasi) adalah pembungkusan sebagai penggabungan potongan-potongan informasi dan perilaku-perilaku yang spesifik yang bekerja pada informasi tersebut, kemudian mengemasnya menjadi object.

```
public class Person {
  private String name; // private = restricted access

  // Getter
  public String getName() {
    return name;
  }

  // Setter
```

```
  public void setName(String newName) {
    this.name = newName;
  }
}

// Penggunaan yang salah
public class Main {
  public static void main(String[] args) {
    Person myObj = new Person();
    myObj.name = "John";  // error
    System.out.println(myObj.name); // error
  }
}
// Penggunaan yang benar
public class Main {
  public static void main(String[] args) {
    Person myObj = new Person();
    myObj.setName("John"); // Set the value of the name variable to
"John"
    System.out.println(myObj.getName());
  }
}

// Outputs "John"
```

g. Class merupakan suatu blueprint atau cetakan untuk menciptakan suatu instance dari object.
h. Class juga merupakan grup suatu object dengan kemiripan attributes/properties, behaviour dan relasi ke object lain.
i. Polymorphism berarti suatu fungsionalitas yang di implementasikan dengan berbagai cara yang berbeda.

   Contoh penggunaan polymorphism pada Java:

```
static int plusMethod(int x, int y) {
  return x + y;
}

static double plusMethod(double x, double y) {
  return x + y;
}

public static void main(String[] args) {
  int myNum1 = plusMethod(8, 5);
  double myNum2 = plusMethod(4.3, 6.26);
  System.out.println("int: " + myNum1);
  System.out.println("double: " + myNum2);
}
//method overloading
```

```
class Animal {
  public void animalSound() {
    System.out.println("The animal makes a sound");
  }
}

class Pig extends Animal {
  public void animalSound() {
    System.out.println("The pig says: wee wee");
  }
}

class Dog extends Animal {
```

```
  public void animalSound() {
    System.out.println("The dog says: bow wow");
  }
}

class Main {
  public static void main(String[] args) {
    Animal myAnimal = new Animal();  // Create a Animal object
    Animal myPig = new Pig();  // Create a Pig object
    Animal myDog = new Dog();  // Create a Dog object
    myAnimal.animalSound();
    myPig.animalSound();
    myDog.animalSound();
  }
}
// Method overriding
```

j. Inheritance bekerja layaknya pewarisan dari orang tua ke anaknya. Inheritance dalam OO berarti mensharing atribut dan behavior antar class dengan hubungan secara hierarki. Fitur pewarisan memungkinkan kita mendefinisikan class baru dengan meng-extend class yang sudah ada.
Contoh penggunaan inheritance pada Java:

```
class Vehicle {
  protected String brand = "Ford";        // Vehicle attribute
  public void honk() {                    // Vehicle method
    System.out.println("Tuut, tuut!");
  }
}

class Car extends Vehicle {
  private String modelName = "Mustang";    // Car attribute
  public static void main(String[] args) {

    // Create a myCar object
    Car myCar = new Car();

    // Call the honk() method (from the Vehicle class) on the myCar
object
    myCar.honk();

    // Display the value of the brand attribute (from the Vehicle
class) and the value of the modelName from the Car class
    System.out.println(myCar.brand + " " + myCar.modelName);
  }
}
```

k. Abstraction adalah proses untuk menyembunyikan beberapa detail dan hanya menunjukkan informasi yang penting saja kepada user.

l. Interface adalah blueprint dari sebuah kelas dan digunakan untuk mengimplementasikan abstraction.
Contoh kode penggunaan interface dalam Bahasa Java

```
interface Animal {
  public void animalSound();
  public void sleep(); // interface method (does not have a body)
}

// Pig "implements" the Animal interface
```

```
class Pig implements Animal {
  public void animalSound() {
    // The body of animalSound() is provided here
    System.out.println("The pig says: wee wee");
  }

  public void sleep() {
    // The body of sleep() is provided here
    System.out.println("Zzz");
  }
}
```

m. Package adalah sekelompok class , interface, dan sub-package dengan tipe yang sama/mirip.

```
package mypack;
class MyPackageClass {
  public static void main(String[] args) {
    System.out.println("This is my package!");
  }
}
```

n. Access specifier adalah hak akses yang diberikan pada method, constructor, class atau attribute yang menentukan sampai sejauh mana komponen-komponen tersebut dapat diakses. Pada java terdapat 4 access specifier, yaitu:
    1. Public, memiliki hak akses paling luas.
    2. Protected, hanya bisa diakses dari child class
    3. Default (no access specifier), dapat diakses selama berada dalam package yang sama
    4. Private, tidak dapat diakses

Access specifier	Class	SubClass	Package	World
Public	√	√	√	√
Protected	√	√	√	X
Default	√	X	√	X
Private	√	X	X	X

F. Aturan dasar penulisan/pembuatan kode program. Untuk dapat memastikan bahwa kode program yang dibuat konsisten, mudah dibaca, dan mudah dipelihara, maka terdapat beberapa aturan dasar yang perlu diperhatikan, yaitu:
    a. Indentasi: Gunakan indentasi yang konsisten (biasanya 2 atau 4 spasi) agar kode menjadi lebih mudah dibaca. Indentasi membantu merepresentasikan blok kode secara visual.
    b. Konvensi Penamaan: Pilih nama yang deskriptif dan bermakna untuk variabel, fungsi, dan kelas.
    c. Tambahkan komentar untuk menjelaskan bagian kode yang kompleks atau proses berpikir yang dilakukan. Komentar harus ringkas namun informatif.
    d. Konsistensi: Tetap berpegang pada gaya pengkodean yang konsisten. Jika sudah ada konvensi untuk bahasa yang digunakan, ikuti konvensi tersebut. Konsistensi dalam gaya kode memudahkan orang lain memahami kode program tersebut.

e. Pecahkan kode menjadi fungsi atau modul atau procedure yang lebih kecil dan dapat digunakan kembali.
f. Menerapkan penanganan kesalahan yang tepat untuk mengelola pengecualian dan kesalahan dengan baik.
g. Gunakan sistem source control (misalnya Git) untuk melacak perubahan pada kode.
h. Tulis pengujian unit dan ikuti prinsip pengembangan berbasis pengujian.
i. Tulis dokumentasi untuk kode program tersebut, termasuk cara menggunakannya, fungsinya, dan ketergantungan apa pun. Ini membantu pengembang lain memahami dan menggunakan kode tersebut.
j. Hindari duplikasi kode; sebagai gantinya, gunakan fungsi atau kelas untuk merangkum fungsionalitas umum.
k. Optimalkan kode untuk kinerja hanya jika diperlukan.

G. Java Overview

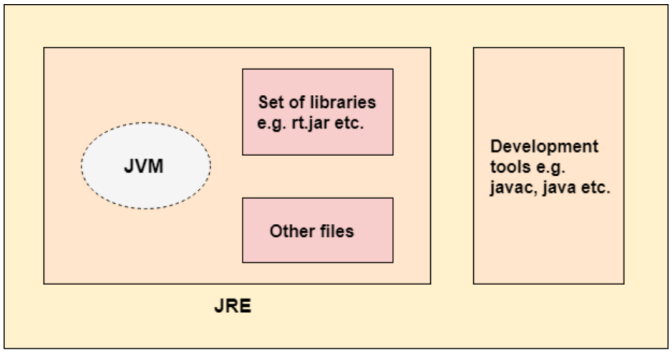
a. Java adalah Bahasa pemrograman popular yang dibuat pada tahun 1995 oleh James Gosling
b. Java adalah bahasa pemrograman tingkat tinggi berorientasi object yang robust dan aman.
c. Perbedaan antara JVM, JRE dan JDK

JVM (Java Virtual Mesin) adalah mesin virtual yang menyediakan lingkungan runtime di mana bytecode Java dapat dieksekusi. JVM yang bertanggung jawab untuk dapat menjalankan program-program yang ditulis dalam Bahasa lain untuk dikompilasi kedalam bentuk bytecode Java.

JRE(Java Runtime Environment) adalah perangkat lunak yang digunakan untuk mengembangkan aplikasi java dan merupakan implementasi dari JVM yang berisi sekumpulan library dan file lainnya yang digunakan JVM pada saat runtime

JDK(Java Development Kit) adalah lingkungan pengembangan perangkat lunak yang digunakan untuk mengembangkan aplikasi dan applet Java.Terdiri dari JRE dan Development Tools.

JavaDoc adalah tools yang disediakan oleh JDK dan digunakan untuk menghasilkan dokumentasi kode Java dalam format HTML dari kode sumber Java yang memerlukan dokumentasi dalam format yang telah ditentukan sebelumnya.



d. IDE yang dapat digunakan untuk mengembangkan Bahasa java adalah Netbeans, eclips, dll

H. Tipe data primitive pada Java

Kategori	Tipe Data	Ukuran Dalam Bit	Default	Rentang Nilai

Integral	byte	8	0	-128	s/d	127
	short	16	0	-32768	s/d	32767
	int	32	0	-2147483648	s/d	2147483647
	long	64	0L	-263	s/d	263-1
Floating Point	float	32	0.0f	1.4e-45	s/d	3.4e+38
	double	64	0.0d	5e-324	s/d	1.8e+308
Textual	char	16	\u0000	\u0000	s/d	\uFFFF
Logical	boolean	1	false	True dan false		

I. Struktur Program dalam Java

a. Pencabangan, dalam java digunakan 2 macam struktur program untuk melakukan percabangan.

```
//Statement If
int x = 20;
int y = 18;
if (x > y) {
  System.out.println("x is greater than y");
}

//Switch
int day = 4;
switch (day) {
  case 6:
    System.out.println("Today is Saturday");
    break;
  case 7:
    System.out.println("Today is Sunday");
    break;
  default:
    System.out.println("Looking forward to the Weekend");
}
// Outputs "Looking forward to the Weekend"
```

b. Perulangan

```
// while
int i = 0;
while (i < 5) {
  System.out.println(i);
  i++;
}

//for
for (int i = 0; i < 5; i++) {
  System.out.println(i);
}

//do … while
int i = 0;
do {
  System.out.println(i);
  i++;
}
while (i < 5);
```

J. Array pada Java

a. Array satu dimensi adalah variabel yang mempunyai nama sama dan tipe data yang sejenis. Untuk membedakan data yang satu dengan yang lainnya dibedakan oleh indeks.

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
cars[0] = "Opel";
System.out.println(cars[0]);
// Now outputs Opel instead of Volvo
```

b. Array multidimensi adalah array yang memiliki lebih dari satu index. Array multidimensi juga dapat dikatakan sebagai array dari array (sekumpulan dari array). Cara mendeklarasikan array multidimensi cukup dengan menambahkan [] sesuai dengan dimensi yang diinginkan.

```
public class Main {
  public static void main(String[] args) {
    int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
    for (int i = 0; i < myNumbers.length; ++i) {
      for(int j = 0; j < myNumbers[i].length; ++j) {
        System.out.println(myNumbers[i][j]);
      }
    }
  }
}
```

K. Prosedur dan Fungsi pada Java

a. Pada bahasa – bahasa pemrograman yang lain method disebut juga fungsi atau prosedur. Dalam pemograman berorientasi object method adalah suatu operasi atau kegiatan yang dapat dilakukan suatu objek. Salah satu kegunaan dari method yaitu digunakan untuk memodularisasi program melalui pemisahan tugas dalam suatu class.

```
class Buku
{
    //Deklarasi variabel
    String judul = "Memoar Sherlock Holmes";
    int total, harga = 45000, jumlah = 2;
    void hitung() //Membuat method bernama hitung()
    {
        total = harga * jumlah;
        System.out.println("Total harga = " + total);
    }
    public static void main(String[] args)
    {
        Buku var = new Buku();
        var.hitung();  //memanggil method
    }
}
```

```
class PersegiPanjang {
  int lebar;
  int tinggi;
  int luasPersegiPanjang() {
    return lebar * tinggi;
  }
}
```

```
public class Main {
  public static void main(String args[]) {
    PersegiPanjang mybox1 = new PersegiPanjang();
    int luas;
    mybox1.lebar = 10;
    mybox1.tinggi = 20;
    luas = mybox1. luasPersegiPanjang();
    System.out.println("Luas Persegi Panjang = " + luas);
  }
}
```

b. Method dengan parameter.

```
public class Buku
{
    String judul = "Memoar Sherlock Holmes";
    int total, harga, jumlah;
    void hitung(int hrg, int jml)
    {
        total = harga * jumlah;
        System.out.println("Total harga = " + total);
    }
    public static void main(String[] args)
    {
        Buku var = new Buku();
        var.harga = 45000;
        var.jumlah = 2;
        System.out.println("Judul = " + var.judul);

        //panggil method dengan mengirimkan parameter
        var.hitung(var.harga, var.jumlah);
    }
}
```

L. Operasi File pada Java

a. FileWriter, class yang digunakan untuk menuliskan teks kedalam file yang telah dibuat sebelumnya. Method yang digunakan adalah method write()

```
      import java.io.FileWriter;   // Import the FileWriter class
      import java.io.IOException;  // Import the IOException class
      to handle errors

      public class WriteToFile {
        public static void main(String[] args) {
          try {
            FileWriter myWriter = new FileWriter("filename.txt");
            myWriter.write("Files in Java might be tricky, but it
      is fun enough!");
            myWriter.close();
            System.out.println("Successfully wrote to the file.");
          } catch (IOException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
          }
        }
}
```

b. FileReader, adalah class yang digunakan untuk membaca isi dari file yang telah ditulis sebelumnya.

```
import java.io.*;
public class ReaderExample {
    public static void main(String[] args) {
        try {
            Reader reader = new FileReader("file.txt");
            int data = reader.read();
            while (data != -1) {
                System.out.print((char) data);
                data = reader.read();
            }
            reader.close();
        } catch (Exception ex) {
            System.out.println(ex.getMessage());
        }
    }
}
```

M. Penanganan Eksepsi dengan Java

a. Kesalahan(error) dalam Java terbagi atas dua kategori yaitu kesalahan pada saat compile (compile time error)dan pada saat runtime (runtime error). Compile-time error terjadi pada saat Anda tidak mengikuti aturan sintaks bahasa pemrograman dalam membuat aplikasi. Compiler akan mendeteksi kesalahan sintaks pada program pada saat compile. Contohnya ketika Anda tidak memasukkan titik koma(;) untuk akhir pernyataan dalam Java. Sedangkan Runtime error adalah kesalahan yang terjadi pada saat program dieksekusi. Inilah yang dimaksud dengan terjadinya eksepsi.

b. Penanganan eksepsi pada java diatur dengan lima kata kunci : try, catch, throw, throws dan finally.

Try, dalam blok ini diletakkan pernyataan yang ingin dimonitor
Catch, blok yang akan dijalankan Ketika eksepsi terjadi sesuai dengan tipe eksepsinya
Throw, digunakan untuk melemparkan eksepsi secara manual
Throws, digunakan untuk melemparkan eksepsi dari method
Finaly, blok kode yang selalu akan dieksekusi, baik Ketika eksepsi terjadi ataupun tidak.

```
class TestFinallyBlock {
  public static void main(String args[]){
  try{
//below code do not throw any exception
   int data=25/5;
   System.out.println(data);
  }
//catch won't be executed
  catch(NullPointerException e){
System.out.println(e);
}
//executed regardless of exception occurred or not
 finally {
System.out.println("finally block is always executed");
}

System.out.println("rest of phe code...");
  }
}
```

N. Akses Database dengan Java

a. Koneksi ke database

```
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
String url =
"jdbc:sqlserver://localhost:1433;databaseName=AdventureWorks";
Connection con =
DriverManager.getConnection(url,"username","password");
```

b. Menampilkan data dari tabel

```
      /import class yang dibutuhkan dalam package java.sql
      import java.sql.*;
      public class Main {
        public static void main(String[] args){
          try {
            //Load Driver JDBC

      Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver")
      ;
            //Bangun Koneksi ke database
            String                      url                      =
      "jdbc:sqlserver://localhost:1433;databaseName=AdventureWorks"
      ;
            Connection                    con                    =
      DriverManager.getConnection(url,"username","password");
            //Buat objek Statement dengan nama stmt
            Statement stmt = con.createStatement();
            //Membuat Query
            String   str   =   "SELECT   DepartmentID,   Name   FROM
      HumanResources.Department";
            ResultSet rs = stmt.executeQuery(str);
            System.out.println("Dep   ID\t   Name");   //\t   untuk
      memisahkan kata dengan tab
            while(rs.next()){
              int depID = rs.getInt("DepartmentID");
              String nama = rs.getString("Name");
              if(depID > 9)
                System.out.println(depID + "\t" + nama);
              else
                System.out.println(depID + "\t " + nama);
            }
            //Tutup koneksi
            con.close();
          }
          catch(Exception ex) {
            System.err.println("Terjadi kesalahan : " + ex);
          }
        }
}
```

c. Memasukkan data ke dalam tabel

```
import java.sql.*;
class InputData {
  public static void main(String[] args) {
    try {

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
      String url = "jdbc:sqlserver://localhost:1433;databaseName=
AdventureWorks";
      Connection con = DriverManager.getConnection(url, "username",
"password");
      Statement stmt = con.createStatement();
      int i = stmt.executeUpdate("INSERT INTO
HumanResources.Department VALUES ('Detektif','Investigasi','2011-
02-14')");
      if(i > 0) {
        System.out.println("Data berhasil dimasukkan sebanyak " + i
+ " rekord");
      }
      con.close();
    }
    catch(Exception ex) {
      System.err.println("Terjadi Error : " + ex);
    }
  }
}
```

d. Mengubah data pada tabel

```
import java.sql.*;
class EditData {
  public static void main(String[] args) {
    try {

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
      String url =
"jdbc:sqlserver://localhost:1433;databasename=AdventureWorks";
      Connection con =
DriverManager.getConnection(url,"username","password");
      Statement stmt = con.createStatement();
      int i = stmt.executeUpdate("UPDATE HumanResources.Department
SET Name = 'Detective' WHERE DepartmentID = 17");
      if(i!=0)
        System.out.println("Berhasil Update");
      else
        System.out.println("Tak ada yang terupdate");
      con.close();
    }
    catch(Exception ex) {
      System.err.println("Terjadi Error : " + ex);
    }
  }
}
```

e. Menghapus data dari tabel

```
import java.sql.*;
class HapusData {
  public static void main(String[] args) {
    try {

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
      String url =
"jdbc:sqlserver://localhost:1433;databaseName=AdventureWorks";
      Connection con =
DriverManager.getConnection(url,"username","password");
      Statement stmt = con.createStatement();
      int i = stmt.executeUpdate("DELETE HumanResources.Department
WHERE DepartmentID = 17");
      if(i!=0)
        System.out.println("Data berhasil dihapus");
      else
        System.out.println("Data tidak terhapus");
      con.close();
    }
    catch(Exception ex) {
      System.err.println("Terjadi Error : " + ex);
    }
  }
}
```

f. Menggunakan object preparedStatment untuk mengubah data pada tabel

```
import java.io.*;
import java.sql.*;
class PreEditData {
  public static void main(String[] args) {
    try {

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
      String url =
"jdbc:sqlserver://localhost:1433;databaseName=AdventureWorks";
      Connection con =
DriverManager.getConnection(url,"username","password");
      BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
      int id;
      String grup;
      System.out.print("Masukkan Department ID yang ingin diubah :
");
      /*Karena setiap nilai yang dimasukkan lewat console berupa
String dan
      disimpan ke dalam variabel bertipe data int maka nilai harus
diparsing
      terlebih dahulu sebelum dimasukkan ke dalam variabel*/
id = Integer.parseInt(br.readLine());
      System.out.print("Masukkan Nama Grup yang baru : ");
      grup = br.readLine();
      String str = "UPDATE HumanResources.Department SET GroupName
= ? WHERE DepartmentID = ?";
```

```
      PreparedStatement pr = con.prepareStatement(str);
      pr.setString(1,grup);
      pr.setInt(2,id);
      int i = pr.executeUpdate();
      if(i!=0)
        System.out.println("Data berhasil diupdate");
      else
        System.out.println("Data tidak ada yang terupdate");
      con.close();
    }
    catch(Exception ex) {
      System.err.println("Terjadi Error : " + ex);
    }
  }
}
```

O. Structured Query Language (SQL)

a. Data Definition Language, mendeskriprikan bagaimana data disimpan dalam database

1. CREATE

```
--Create Database
CREATE DATABASE Rekening_Nasabah

--Create Table
CREATE TABLE Nasabah
(
Id_nasabah Char   (6) NOT NULL PRIMARY KEY,
Nm_nasabah Varchar (50) NOT NULL,
Alamat      Varchar     (50) NOT NULL,
Jns_kelamin Char (1) NOT NULL,
No_telp     Char  (10) NOT NULL,
Status      Varchar     (20) NOT NULL,
Agama Varchar     (15) NOT NULL,
Pekerjaan   Varchar     (50) NOT NULL,
Nama_ibu_kandung Varchar (50) NOT NULL
)
```

2. DROP

```
--drop database
DROP DATABASE Employee

--drop table
Drop Table Nasabah
```

3. ALTER

```
ALTER TABLE DataNasabah.Nasabah
ADD KD_Pos CHAR (5) NOT NULL
```

b. Data Manipulation Language

1. SELECT

```
SELECT * FROM HumanResources.Employee WHERE
Title = 'Marketing Manager' AND MaritalStatus = 'S'
```

2. INSERT

```
(Id_nasabah,Nm_Nasabah,Alamat,Jns_Kelamin,No_Telp,Status,Ag
ama,Pekerjaan,Nama_ibu_kandung)
VALUES
('nsb001','Supian','Depok','L','0219101010','Menikah','Isla
m','Karyawan','Sundari')
```

3. UPDATE

```
UPDATE DataNasabah.Nasabah
SET Status = 'Menikah' WHERE Id_nasabah = 'nsb002'
```

4. DELETE

```
DELETE DataNasabah.Kartu_Identitas
WHERE Id_nasabah = 'nsb002'
```

c. Membuat Procedure dan Trigger

1. Procedure adalah sekelompok perintah sql yang disimpan dengan sebuah nama sehingga dapat digunakan kembali kapan saja dengan cara mengeksekusinya

```
CREATE PROCEDURE procNasabahAlamat @Alamat varchar(50)
AS
BEGIN
PRINT 'Daftar Nasabah'
SELECT Id_Nasabah, Nm_Nasabah,Alamat FROM
DataNasabah.Nasabah
WHERE Alamat=@Alamat
END

EXEC procNasabahAlamat 'Depok'


CREATE PROCEDURE procUpdateNasabah @IDNasabah char(6),
@Alamat varchar(50), @NoTelp char(10), @Pekerjaan
varchar(50)
AS
BEGIN
UPDATE DataNasabah.Nasabah
SET Alamat=@Alamat, No_telp=@NoTelp, Pekerjaan=@Pekerjaan
WHERE Id_nasabah=@IDNasabah
END

EXEC procUpdateNasabah
'nsb001','Bekasi','(021)7777777','Karyawan Swasta'
```

2. Trigger, adalah object database yang terdiri atas sekelompok perintah SQL yang akan dijalankan otomatis setelah perintah tertentu dijalankan

```
CREATE TRIGGER trgInsertRek
ON DataRekening.Rekening
FOR INSERT
AS
DECLARE @tgl_buka datetime
SELECT @tgl_buka = Tgl_buka FROM Inserted
IF (@tgl_buka != getdate())
BEGIN
PRINT 'Tanggal Pembukaan Rekening harus sesuai dengan
tanggal sekarang, maka data tidak dapat disimpan.'
ROLLBACK TRANSACTION
END
RETURN

--triger update
CREATE TRIGGER trgUpdateSaldo
ON DataRekening.Rekening
FOR UPDATE
AS
IF UPDATE (Saldo)
BEGIN
DECLARE @Saldo money
DECLARE @Id_Tabungan char(6)
DECLARE @Saldo_min money

SELECT @Id_Tabungan = Id_Tabungan, @Saldo = Saldo FROM
Inserted

SELECT @Saldo_min = Saldo_min FROM
DataRekening.Jenis_Tabungan
WHERE Id_tabungan =  @Id_Tabungan
IF(@Saldo <  @Saldo_min)
BEGIN
PRINT 'Nilai Saldo lebih kecil dari Saldo Minimum'
ROLLBACK TRANSACTION
END
            END
```

P. Pengujian

a. Strategi Pengujian

1. Blackbox Testing, pengujian yang dilakukan hanya untuk memastikan fungsionalitas dari perangkat lunak sudah benar tanpa mempertimbangakan bagaimana perangkat lunak dibuat
2. Whitebox Testing, pengujian yang detail terhadap kode program, fungsi, prosedur, loop, kondisi. Sehingga diperlukan pengetahuan yang khusus untuk memahami kode program.

b. Level Pengujian

1. Unit Testing, adalah pengujian yang dilakukan pada unit terkecil perangkat lunak untuk memastikan tidak ada kesalahan/ error atau bug dan fungsi dari unit tersebut

telah berjalan dengan baik. Unit terkecil ini bisa saja berupa method dari sebuah class, class, sekolompok inter-related class, atau module.Unit testing dilakukan oleh developer.

2. Integration Testing, adalah pengujian yang dilakukan untuk memeriksa apakah tiap-tiap unit yang disatukan dapat beroperasi dengan baik.
3. System Testing, adalah pengujan yang dilakukan untuk memeriksa apaka keselurah sistem perangkat lunak telah sesuai dari sisi fungsionalitasnya.
4. Regression Testing, pengujian yang dilakukan untuk melihat apakah ada efek//ror terhadap perangkat lunak setelah dilakukan modifikasi.
5. Acceptance Testing, adalah pengujian yang dilakukan denganmengkomparasi requirement awal customer terhadap perangakat lunak yang dihasilkan
6. Performance Testing, adalah pengujian yang dilakukan untuk menguji efisiensi dan kecepatan perangkat lunak mengeksekusi tugasnya.

Daftar Pustaka

https://codepolitan.com/blog/mengenal-diagram-uml-unified-modeling-language/

https://www.mahirkoding.com/mengenal-access-modifier-pada-bahasa-java/

https://www.w3schools.com/java/

https://www.javatpoint.com/access-modifiers

https://www.researchgate.net/figure/State-Transition-Diagram-for-the-Elevator-Controller_fig1_267767796

https://www.tutorialspoint.com/java/

https://wiki.edunitas.com/IT/114-10/Component-Object-Model_1341_eduNitas.html#COM.2B

https://courses.cs.washington.edu/courses/csep503/98au/ShamikFinal.htm

https://wiki.edunitas.com/IT/114-10/DCOM_1641_eduNitas.html

https://www.tutorialspoint.com/webservices/what_are_web_services.htm