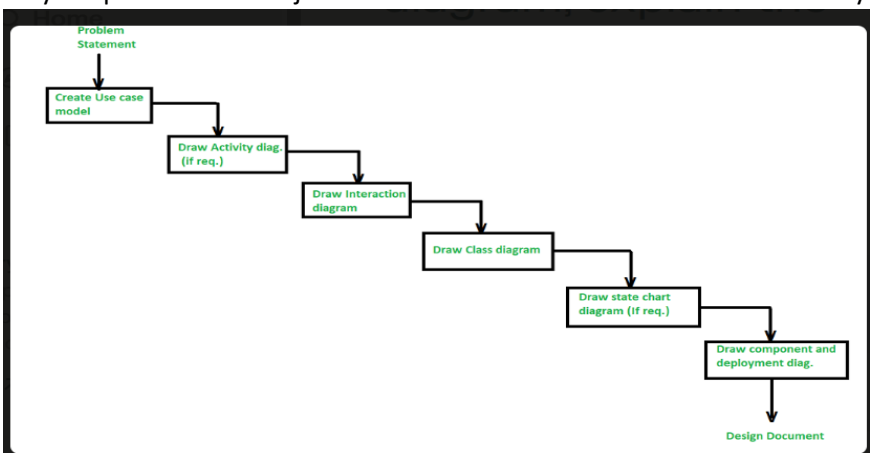
KIMATHI EVALOISE KAIMURI
SCT1211011-2022
OOP ASSIGNMENT

PART A:

i. Creating a System Using OOP Principles:

i. Identify Objects: Identify the objects relevant to the system.
ii. Define Classes: Create classes for each identified object.
iii. Define Attributes: Assign attributes (properties) to each class.
iv. Define Behaviors: Add methods (functions) to implement behaviors.
v. Encapsulation: Encapsulate the attributes and methods within classes.
vi. Inheritance: Establish relationships and hierarchies between classes.
vii. Polymorphism: Allow objects of different classes to be treated uniformly.



ii. Object Modeling Techniques (OMT):

Object Modeling Techniques (OMT) is a method for object-oriented analysis and design that uses graphical notation to depict the objects, classes, and their relationships.

iii. OOAD vs. OOP:

Object-Oriented Analysis and Design (OOAD): It's a software methodology that involves using Object oriented concept to design and implement software system.

Object Analysis and Design (OOP) is basically a computer programming design philosophy or methodology that organizes/ models software design around data, or objects rather than functions and logic.

iv. Main Goals of UML:

- ✓ Provide a standardized modeling language.
- ✓ Support visual representation of system structure and behavior.
- ✓ Facilitate communication and understanding among stakeholders.

v. Advantages of Object-Oriented Development:

- ➢ Modularity: Code is organized into manageable, independent modules.

KIMATHI EVALOISE KAIMURI
SCT1211011-2022
OOP ASSIGNMENT

- Reusability: Classes and objects can be reused in different parts of the system.
- Maintainability: Changes in one part of the system do not affect others.

vi. Explanation of Terms in OOP:

a. **Constructor:** Initializes an object when it is created.

```
public class Person {
   private String name;
  private int age;
   public Person(String name, int age) {
      this.name = name;
      this.age = age;
   }
}
```

b. **Object:** An instance of a class.

```
Person person = new Person("John", 30);
```

c. **Polymorphism** is the ability of an object to take on many forms.

```
public class Animal {
   public void makeSound() {
      System.out.println("The animal makes a sound");
   }
}
public class Dog extends Animal {
   @Override
   public void makeSound() {
      System.out.println("The dog barks");
   }
}
public class Cat extends Animal {
   @Override
   public void makeSound() {
      System.out.println("The cat meows");
   }
}
public class Main {
   public static void main(String[] args) {
      Animal animal1 = new Dog();
      Animal animal2 = new Cat();

      animal1.makeSound(); // Output: The dog barks
      animal2.makeSound(); // Output: The cat meows
   }
}
```

d. **Class:** A class is a blueprint for creating objects.

```
public class Person {
```

KIMATHI EVALOISE KAIMURI
SCT1211011-2022
OOP ASSIGNMENT

```
    private String name;
    private int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public String getName() {
        return name;
    }
    public int getAge() {
        return age;
    }
}
```

e) Destrutor is not used in Java

f. Inheritance: Inheritance is the ability of a class to inherit properties and methods from its superclasspublic class Animal {

```
    public void makeSound() {
        System.out.println("The animal makes a sound");
    }
}
public class Dog extends Animal {
    public void bark() {
        System.out.println("The dog barks");
    }
}
```

vii. Three Types of Associations in OOP:

- Aggregation: Weaker relationship; objects can exist independently.
- Composition: Stronger relationship; one object owns another.
- Association: General relationship; one class is related to another.

viii. Class Diagram:

A **class diagram** is a visual representation of classes, relationships, and their associations in a system.

It is Used in UML to model the static view of a system.

Steps to Draw:

1. Identify classes.
2. Determine relationships and associations.
3. Add attributes and methods.
4. Draw connections between classes.

vii. Inheritance, Friend Functions, Method Overloading, Late Binding, Abstract Class:

viii. Differentiation in C++:

a. Function Overloading and Operator Overloading:

KIMATHI EVALOISE KAIMURI
SCT1211011-2022
OOP ASSIGNMENT

Function Overloading: Multiple functions with the same name but different parameters or types.

Operator Overloading: Redefining operators for user-defined types. In the Complex class, + operator is overloaded.