“UNIVERSIDAD CENTRAL DEL ECUADOR”

Topic: Investigation.

Participants: Kevin Segovia, Leidy Vacacela.

Date: 2024/06

Teacher: Juan Pablo Guevara

Contents

UML Diagrams

UML diagram is a way to visualize systems and software using Unified Modeling Language (UML). Software engineers create UML diagrams to understand the designs, code architecture, and proposed implementation of complex software systems. UML diagrams are also used to model workflows and business processes.

Coding can be a complicated process with many interrelated elements. There are often thousands of lines of programming language that can be difficult to understand at first glance. A UML diagram simplifies this information into a visual reference that’s easier to digest. It uses a standardized method for writing a system model and capturing conceptual ideas.

-An interface is a completely "abstract class" that is used to group related methods with empty bodies

To access the interface methods, the interface must be "implemented" (like inherited) by another class with the implements keyword (instead of extends). The body of the interface method is provided by the "implement" class

UML Diagrams: Types

There are two subcategories of UML diagrams: structural diagrams and behavioral diagrams.

Structural diagrams depict the components that make up a system and the relationship between those components. These diagrams show the static aspects of a system.Behavioral diagrams represent what happens within a system. They show how all the components interact with each other and with other systems or users.

Class diagram: A UML class diagram is a fundamental building block of any object-oriented solution. It depicts a static, object-oriented system, defining projects by classes, attributes, and functions.

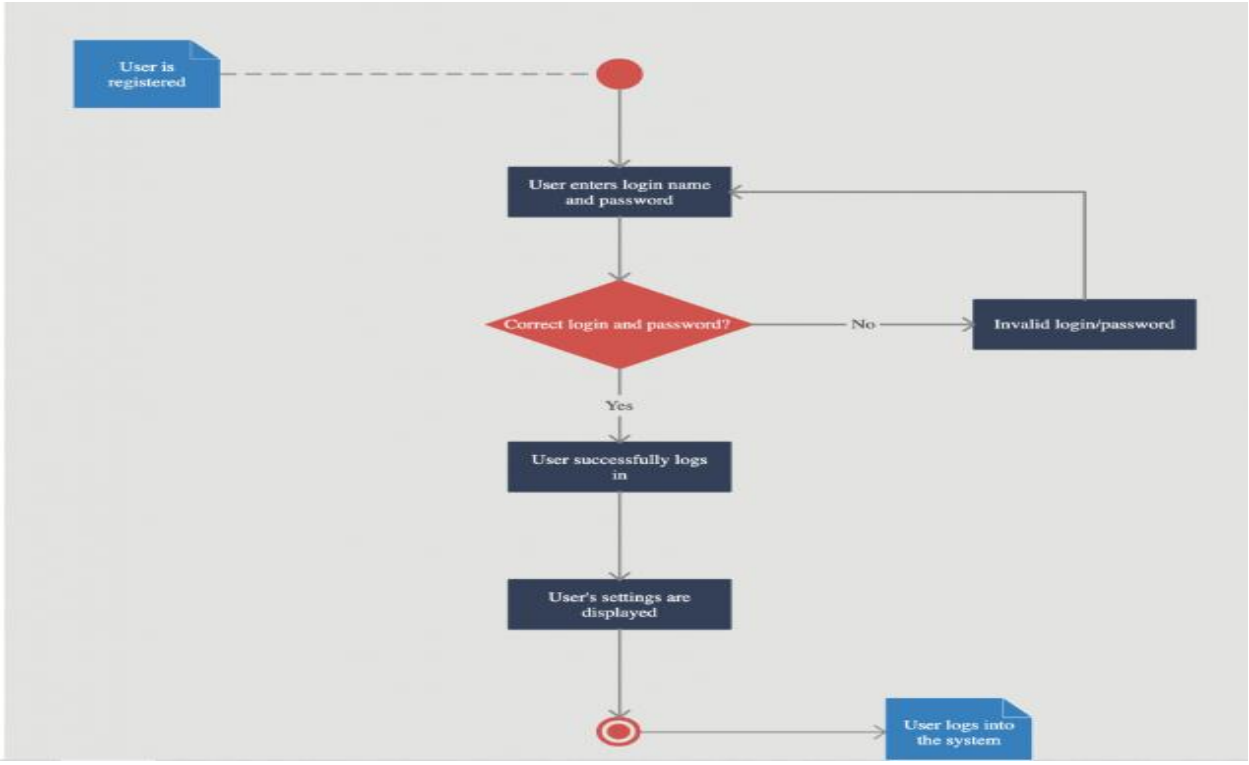
Package diagram: As the name suggests, a package diagram shows the dependencies and relationships between packages in a system.

Activity diagram: A UML activity diagram is a flowchart that outlines all of a system's activities.

Sequence diagram: UML sequence diagrams — sometimes known as event diagrams — show the order in which your objects interact.

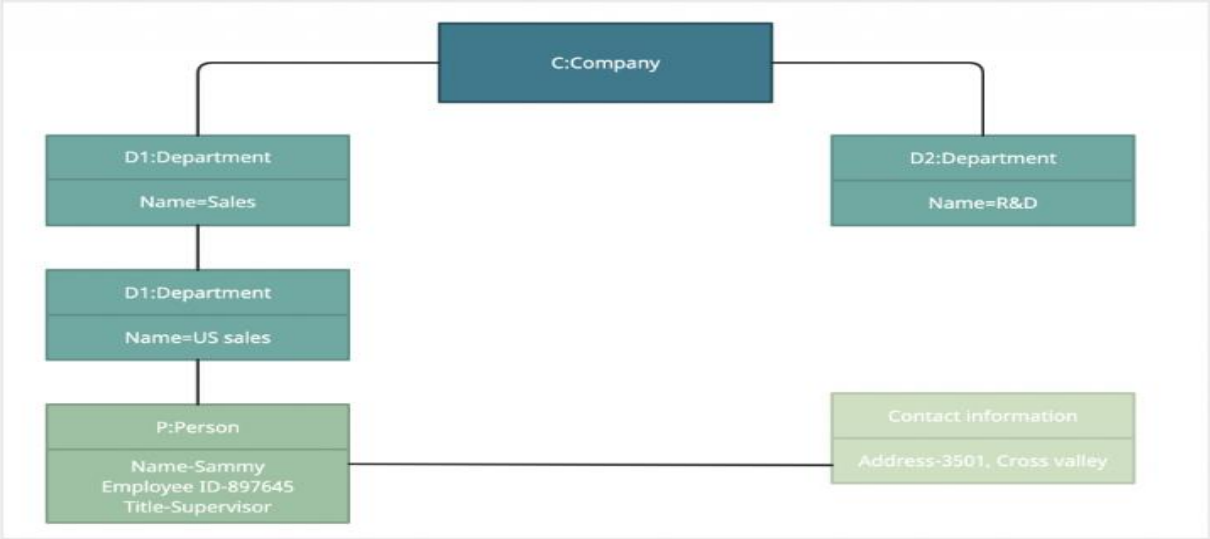
UML Diagrams: Examples

Activity diagrams represent workflows in a graphical way. They can be used to describe the business workflow or the operational workflow of any component in a system.



Shape 1.

Object Diagrams, sometimes referred to as Instance diagrams are very similar to class diagrams. Like class diagrams, they also show the relationship between objects but they use real-world examples.



Shape 2.

Interface

An interface is a completely "abstract class" that is used to group related methods with empty bodies

To access the interface methods, the interface must be "implemented" (like inherited) by another class with the implements keyword (instead of extends). The body of the interface method is provided by the "implement" class

Like abstract classes, interfaces cannot be used to create objects

Interface methods do not have a body - the body is provided by the "implement" class

On implementation of an interface, you must override all of its methods

Interface methods are by default abstract and public

Interface attributes are by default public, static and final

An interface cannot contain a constructor (as it cannot be used to create objects)

Why And When To Use Interfaces?

1) To achieve security - hide certain details and only show the important details of an object (interface).

2) Java does not support "multiple inheritance" (a class can only inherit from one superclass). However, it can be achieved with interfaces, because the class can implement multiple interfaces.-

Abstract class

Data abstraction is the process of hiding certain details and showing only essential information to the user.

Abstraction can be achieved with either abstract classes or interfaces (which you will learn more about in the next chapter).

The abstract keyword is a non-access modifier, used for classes and methods:

Abstract class: is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).

Abstract method: can only be used in an abstract class, and it does not have a body. The body is provided by the subclass (inherited from).

An abstract class can have both abstract and regular methods

To access the abstract class, it must be inherited from another class.

Why And When To Use Abstract Classes and Methods?

To achieve security - hide certain details and only show the important details of an object.

Inheritance

Java Inheritance (Subclass and Superclass)

In Java, it is possible to inherit attributes and methods from one class to another. We group the "inheritance concept" into two categories:

Subclass (child) - the class that inherits from another class

Superclass (parent) - the class being inherited from

To inherit from a class, use the extends keyword.

Why And When To Use "Inheritance"?

- It is useful for code reusability: reuse attributes and methods of an existing class when you create a new class.

```
1  abstract class Brand {
2          abstract void name();
3          void Cost(){
4              System.out.println("100$");
5          }
6      }
```

Shape 3.

```java
public class Logitech extends Mouse {
    public void dpi(){
        System.out.println("max 10000 dpi");
    }
    public void cost(){
        System.out.println("The logitech g502 cost 100 dollars");
    }
}
```

Shape 4.

```java
public class Main {
    public static void main(String[] args) {
    brandShoes mybrandShoes = new brandShoes();
    mybrandShoes.name();
    mybrandShoes.Cost();
    Logitech myLogitech = new Logitech();
    myLogitech.cost();
    myLogitech.dpi();
    }
}
```

Shape 5.

```java
abstract class Mouse{
    abstract void dpi();
    abstract void cost();
}
```

Shape 6.

```java
class brandShoes extends Brand {
    public void name(){
        System.out.println("Adidas");
    }
}
```

Shape 7.

Bibliography

Eriksson, H.-E., & Penker, M. (s/f). *Business Modeling with UML*. Utm.mx. Recuperado el 28 de junio de 2024, de https://www.utm.mx/~caff/poo2/Business%20Modeling%20with%20UML.pdf

Java abstraction. (s/f). W3schools.com. Recuperado el 28 de junio de 2024, de https://www.w3schools.com/java/java_abstract.asp

Java inheritance. (s/f). W3schools.com. Recuperado el 28 de junio de 2024, de https://www.w3schools.com/java/java_inheritance.asp

Java interface. (s/f). W3schools.com. Recuperado el 28 de junio de 2024, de https://www.w3schools.com/java/java_interface.asp

Nishadha. (2012, febrero 1). *UML diagram types guide: Learn about all types of UML diagrams with examples*. Creately Blog; Creately. https://creately.com/blog/diagrams/uml-diagram-types-examples/

What is a UML diagram? (s/f). Https://miro.com/. Recuperado el 28 de junio de 2024, de https://miro.com/diagramming/what-is-a-uml-diagram/

Reggio, G., Leotta, M., Ricca, F., & Clerissi, D. (2013, October). What are the used UML diagrams? A Preliminary Survey. In *EESSMod@ MoDELS* (pp. 3-12).