QUIZ 1

(OBJECT ORIENTED PROGRAM)

ANANDA AZ HARUDDIN SALIMA

2241720071

Class dan object:

1. What is a "class" in object-oriented programming?

   A class is an entity that defines the attributes (variables) and methods (functions) that an object created based on that class will have. The concept of classes allows us to organize and group code into larger units.

2. How do you define the object of a class in Java?

   First, you must define a class by using the "class" keyword. Example:

```
public class Barang {
// Atribut dan metode akan didefinisikan di sini.
}
```

   Next, you can create an object from that class by using the "new" statement. Example:

```
public class Barang {
// Atribut dan metode akan didefinisikan di sini.
Barang laptop = new Barang();
}
```

1. Suppose you have a class of "Goods" in an inventory information system. How would you create a "laptop" object from that class?

```
public class Barang {
    String nama;
    int jumlah;
    public Barang(String nama, int jumlah) {
        this.nama = nama;
        this.jumlah = jumlah;
    }
}
public class Main {
    public static void main(String[] args) {
        Barang laptop = new Barang("Laptop Acer", 5);
    }
}
```

1. Encapsulation:
1. Explain the concept of encapsulation in object-oriented programming and why it is important in the development of inventory information systems.

   In programming, this means that the attributes (variables) and methods (functions) associated with an object must be stored in a class and accessible through a predefined interface, i.e. a method defined in that class.

   1. Data Security: Encapsulation allows you to avoid direct access to certain features. This helps prevent unauthorized or unwanted changes to inventory data, which can compromise system integrity and consistency, by indicating that object data can only be changed in a predefined manner.

   2. Abstraction: Encapsulation makes it easier for interfaces to interact with objects. Users of inventory information systems only need to know how to use predefined methods to interact with objects, and they do not need to know internal details such as data structures and algorithms.

3. Flexibility: Encapsulation allows you to change the internal implementation of a class or improve it without interfering with the use of classes in other parts of the system. This allows upgrading the system without interrupting the use of existing objects.

1. In the context of inventory information systems, mention examples of attributes (variables) that should be encapsulate and why.

Examples of attributes that should be encapsulate in an inventory information system are:

1. Price of Goods: The price of goods is sensitive information that must be protected, so it should be encapsulate. With encapsulation, you can apply validation to ensure that prices always have the right value and control who can access or change them.

2. Total Stock: This feature must be encapsulate because changes in the amount of stock items must be done carefully. You should ensure that stock additions or subtractions are only done in a predetermined manner to avoid incorrect counting issues or negative stock.

3. Item Name and Description: To prevent unauthorized or unwanted alteration, this information must also be encapsulate. By providing the right access methods, you can ensure that data is always consistent.

1. Class Relations:
1. What are the relationships between classes in object-oriented programming?

Relationships between classes in object-oriented programming refer to how two or more classes interact or relate to each other in a system.

2. In an inventory information system, how would you describe the relationship between the "Goods" class and the "Category" class?

"Category" is a class that represents a type or category of goods, such as "Electronics," "Clothing," or "Kitchen Appliance."

Goods are classes that represent individual items that are in inventory. Each item must be associated with one particular category to group it into relevant categories.

3. PBL:
1. Based on the case of an inventory information system, try creating a simple class with its attributes and methods that describes an entity in that system (for example, the class "Goods").

```
public class Barang {
    private String kodeBarang;
    private String nama;
    private double harga;
    private int stok;
    public Barang(String kodeBarang, String nama, double harga, int stok) {
        this.kodeBarang = kodeBarang;
        this.nama = nama;
        this.harga = harga;
        this.stok = stok;
```

```
    }
    public String getKodeBarang() {
        return kodeBarang;
    }
    public String getNama() {
        return nama;
    }
    public double getHarga() {
        return harga;
    }
    public int getStok() {
        return stok;
    }
    public void tambahStok(int jumlah) {
        stok += jumlah;
    }
    public void kurangiStok(int jumlah) {
        if (stok >= jumlah) {
            stok -= jumlah;
        } else {
            System.out.println("Stok tidak mencukupi.");
        }
    }
}
```

1. How would you use encapsulation to protect attributesattributes in the class?

1. The "Itemcode," "name," "price," and "stock" attributes are declared as private, so they can only be accessed from within the "Item" class itself. This protects those attributes from unauthorized changes from outside the class.

2. The "getItemCode," "getName," "getPrice," and "getStok" methods are used to grant read-only access to these attributes. This allows the user of the class to see its value without being able to change it directly.
3. The "addStock" and "subtract" methods are used to regulate how stock changes are made, and they check if the changes are valid before changing the stock value. This ensures that the stock cannot become negative.

4. Describe class hierarchies or relationships between classes that may exist in the inventory information system in the Information Technology department. Give examples of relationships between classes (for example, inheritance or association) in that context.

1. Class "Goods" that represent individual items in inventory.

2. The "ItemCategory" class can be used to group items into specific categories.

3. Class "Transactions" that can be used to record sales or purchases of goods.

4. A "User" class that represents a system user who can perform inventory-related operations, such as adding, changing, or deleting items.

Examples of relationships between classes:

1. The classes "Goods" and "Categories of Goods" can have an association relationship in which each item is related to one particular category of goods.

2. The class "Transaction" can have an association relationship with "Goods" to record the goods involved in the transaction.

3. The class "User" can have an association relationship with "Goods" to identify operations permitted by the user to items in inventory.