Java OOPS:

Java is an object-oriented programming (OOP) language, which means it follows the principles of object-oriented programming. OOP is a programming paradigm that uses objects and classes to organize and structure code. Here are some fundamental concepts of OOP in Java, along with code examples for each one:

1. Classes and Objects:
- **Classes** define blueprints or templates for objects, specifying their attributes (fields) and behaviors (methods).
- **Objects** are instances of classes, representing real-world entities.

```java
// Example of a class
class Car {
    String brand;
    int year;

    void startEngine() {
        System.out.println("Engine started.");
    }
}

// Creating objects from the class
Car myCar = new Car();
myCar.brand = "Toyota";
myCar.year = 2022;
myCar.startEngine();
```

2. Encapsulation:
- **Encapsulation** is the practice of hiding the internal details of an object and providing access to its data through methods.
- Access modifiers like `public`, `private`, and `protected` control access to fields and methods.

```java
class BankAccount {
    private double balance;

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
        }
    }

    public double getBalance() {
        return balance;
    }
}
```

3. Inheritance:
   - **Inheritance** allows a class (subclass or derived class) to inherit properties and behaviors from another class (superclass or base class).
   - It promotes code reuse and hierarchy.

```java
    class Animal {
        void eat() {
            System.out.println("Animal is eating.");
        }
    }

    class Dog extends Animal {
        void bark() {
            System.out.println("Dog is barking.");
        }
    }
```

4. Polymorphism:
   - **Polymorphism** allows objects of different classes to be treated as objects of a common superclass.
   - It is achieved through method overriding and interfaces.

```java
    interface Shape {
        void draw();
    }

    class Circle implements Shape {
        @Override
        public void draw() {
            System.out.println("Drawing a circle.");
        }
    }
```

```
class Square implements Shape {
    @Override
    public void draw() {
        System.out.println("Drawing a square.");
    }
}
```

5. Abstraction:
   - **Abstraction** involves simplifying complex systems by breaking them into smaller, more manageable parts.
   - Abstract classes and methods define a common interface without providing implementation details.

```java
abstract class Shape {
    abstract void draw();
}

class Circle extends Shape {
    @Override
    void draw() {
        System.out.println("Drawing a circle.");
    }
}

class Square extends Shape {
    @Override
    void draw() {
        System.out.println("Drawing a square.");
    }
}
```

These are the core concepts of object-oriented programming in Java. They enable developers to create well-structured, modular, and maintainable code by modeling real-world entities as objects and using principles like encapsulation, inheritance, polymorphism, and abstraction to manage complexity and promote code reusability.