Object-oriented programming (OOP) in Java offers several advantages for software development. Here are ten of them, along with code examples to illustrate each advantage:

1. Code Reusability:
OOP promotes code reuse through inheritance and composition.

```java
class Vehicle {
    void start() {
        System.out.println("Vehicle started.");
    }
}

class Car extends Vehicle {
    void drive() {
        System.out.println("Car is driving.");
    }
}

class Bike extends Vehicle {
    void ride() {
        System.out.println("Bike is riding.");
    }
}
```

2. Encapsulation:
OOP encapsulates data and behavior within objects, hiding internal details.

```java
class BankAccount {
    private double balance;

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

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

3. Modularity:
OOP promotes code organization into classes and modules, making it easier to manage and maintain large codebases.

```java
// Different classes for different responsibilities
class Order {
    // ...
}

class Customer {
    // ...
}

class Product {
    // ...
}
```

4. Polymorphism:
OOP allows objects of different classes to be treated as objects of a common superclass, enabling flexibility and extensibility.

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

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

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

5. Inheritance:
OOP supports the creation of new classes by inheriting properties and behaviors from existing classes.

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

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

6. Flexibility and Extensibility:
OOP allows easy modification and extension of existing classes without affecting the overall system.

```java
class Circle extends Shape {
    double radius;

    Circle(double radius) {
        this.radius = radius;
    }

    @Override
    void draw() {
        System.out.println("Drawing a circle with radius " + radius);
    }
}
```

7. Maintainability:
OOP promotes code organization and reduces complexity, making it easier to maintain and debug code.

```java
// Clear separation of concerns
class Customer {
```

```
    // ...
}

class Order {
    // ...
}
```

8. Abstraction:
OOP allows modeling complex systems by defining abstract classes and methods.

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

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

9. Security:
OOP provides access control mechanisms, allowing you to restrict access to certain data and methods.

```java
class SecureData {
    private String secret;

    // Methods to access secret data with proper validation
    // ...
}
```

10. Collaboration:
OOP promotes collaboration among development teams by defining clear interfaces between classes.

```java
// Interfaces define contracts
interface PaymentGateway {
```

```
    void processPayment(double amount);
}

class PayPal implements PaymentGateway {
    @Override
    public void processPayment(double amount) {
        // Implement PayPal payment logic
    }
}
```

These advantages make OOP in Java a powerful and widely used approach for designing and building software systems that are modular, maintainable, and extensible.