DR MOJTABA VAHIDI ASL

Expert OOP & Interface

Presenters:

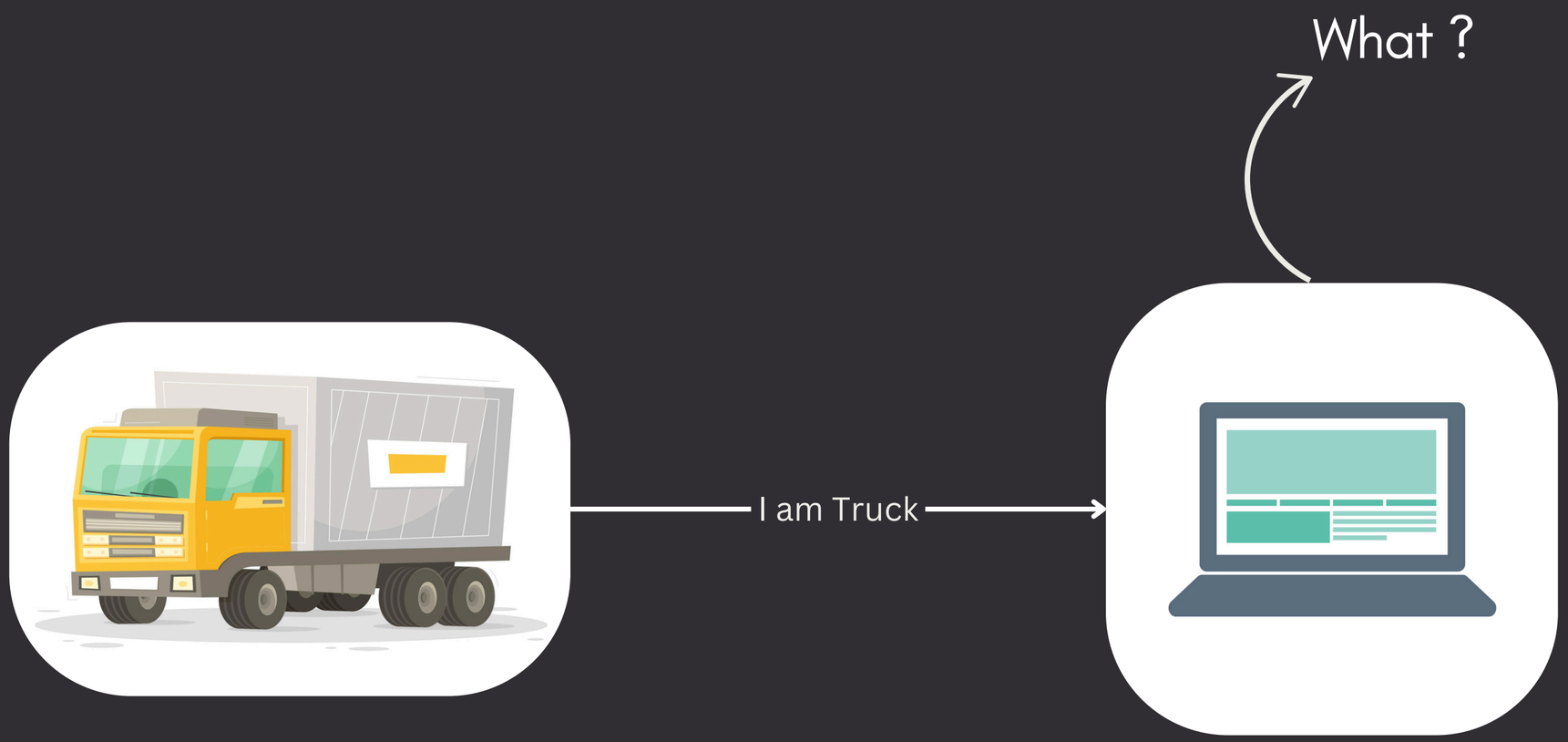
Amirmahdi Mashayekhi
telegram id : @mashmool0
telegram channel : @mashmoolcode

Date:

1404 AP Spring

Review OOP

What is Object-Oriented Programming?


What ?
I am Truck

```java
public class Truck {
    private String plateNumber;
    private String model;
    private int manufactureYear;
    private int numberOfTires;
    private Date[] lastTireChangeDate;
}
```

Cars

Inheritance

```java
public class Car {
    // Protected fields accessible by child classes
    protected String brand;
    protected String model;
    protected int year;
    protected String plateNumber;
    protected String color;
    protected int numberOfWheels;

}

// Truck class inheriting from Car
public class Truck extends Car {
    // Additional truck-specific fields
    private double cargoCapacity;
    private String truckType; // e.g., "Semi", "Pickup", "Dump truck"
    private boolean hasTrailer;
    private Date[] tireChangeDates;
}
```

Code Reusability

Weight Of Different Car

Polymorphism

```java
public class Car {
    protected String brand;
    protected String model;
    protected int year;
    protected int numberOfWheels;

    public int calculateTirePressure(int weight) {
        return weight / numberOfWheels;
    }
}

public class DumpTruck extends Car {
    private int maxLoad;

    // Constructor

    @Override
    public int calculateTirePressure(int weight) {
        // فشار بیشتر به دلیل حمل بار سنگین
        return (weight / numberOfWheels) + 20;
    }
}

public class CementTruck extends Car {
    private boolean isMixerOn;


    // Constructor

    @Override
    public int calculateTirePressure(int weight) {
        // فشار اضافی به خاطر چرخش میکسر
        int basePressure = weight / numberOfWheels;
        return isMixerOn ? basePressure + 35 : basePressure + 15;
    }
}
```

Your Example

Encapsulation

Bad Example

```java
class Person {
    private String name;
    private int age;

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public int getAge() { return age; }

    public void setAge(int age) { this.age = age; }
}
```

Why encapsulation?

Security

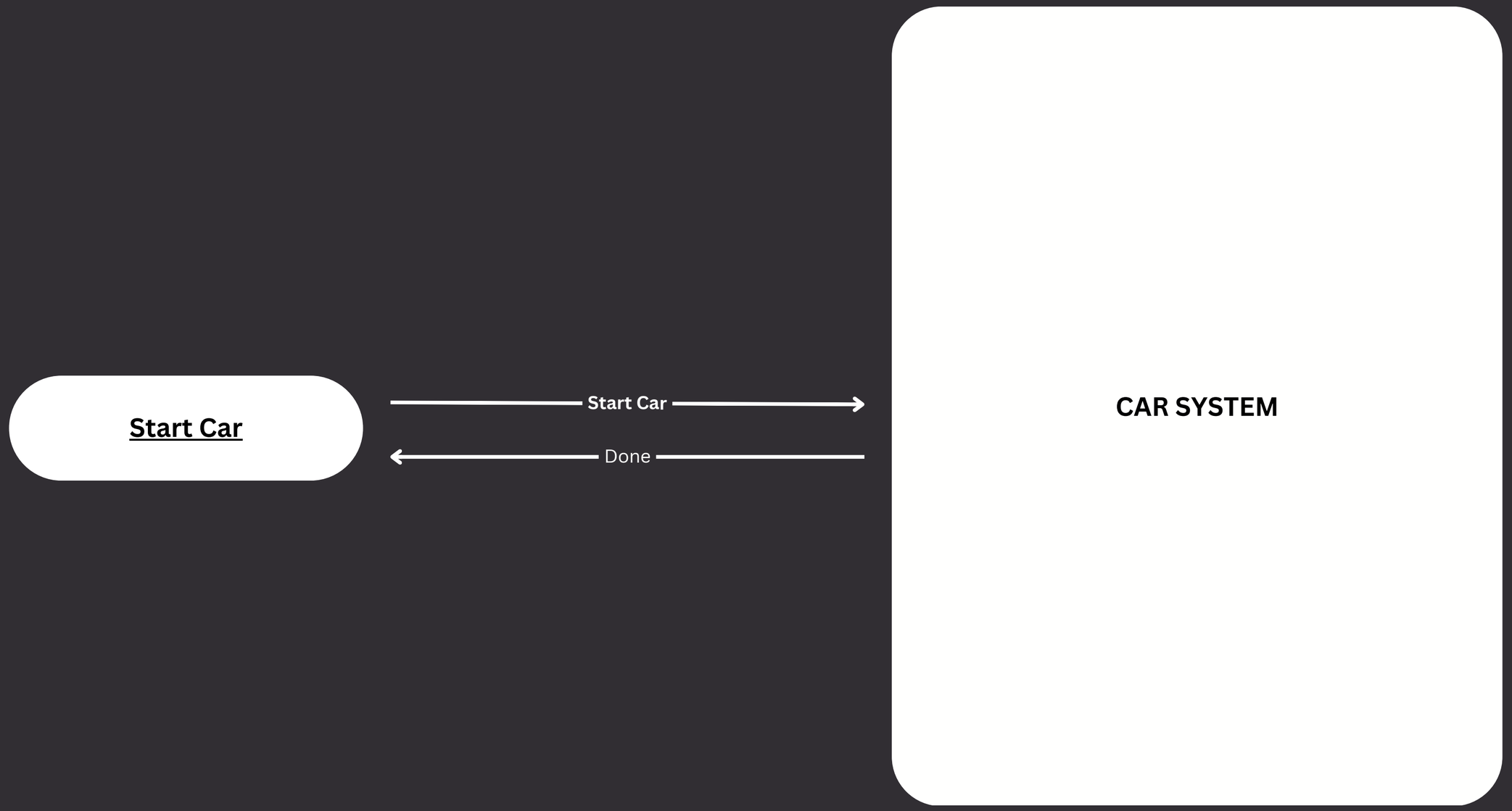
```java
public class CreditCard {
    private String cardNumber;
    private String cvv;
    private String expirationDate;
    private double balance;
    private int wrongPinAttempts;
    private boolean isBlocked;
    private String pin;

    public String getCardNumber() {
        // فقط 4 رقم آخر کارت رو نمایش میده
        return "****-****-****-" + cardNumber.substring(cardNumber.length() - 4);
    }
```

Flexibility

```java
public class Employee {
    private String name;
    private double salary;

    public double getSalary() {
        return calculateSalaryWithBonus();
    }

    private double calculateSalaryWithBonus() {
        return salary + (salary * 0.1);
    }
}
```

Q&A

Abstraction


Start Car
Start Car
Done
CAR SYSTEM

Start Car

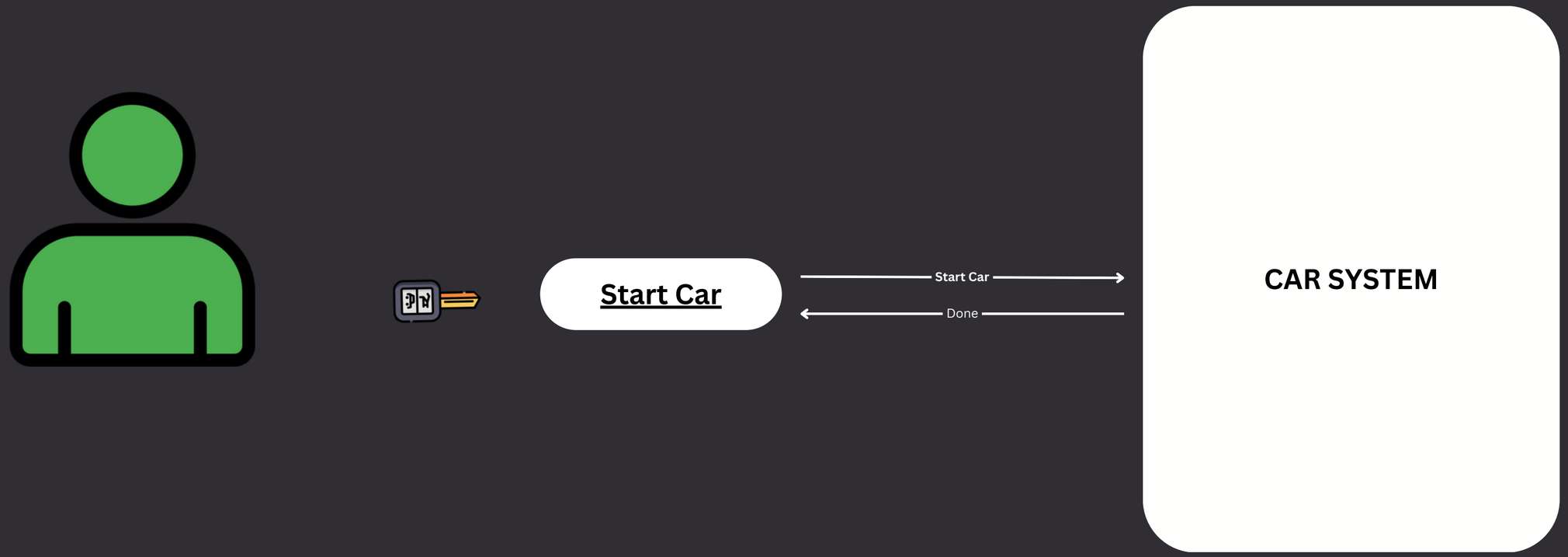
Start Car →

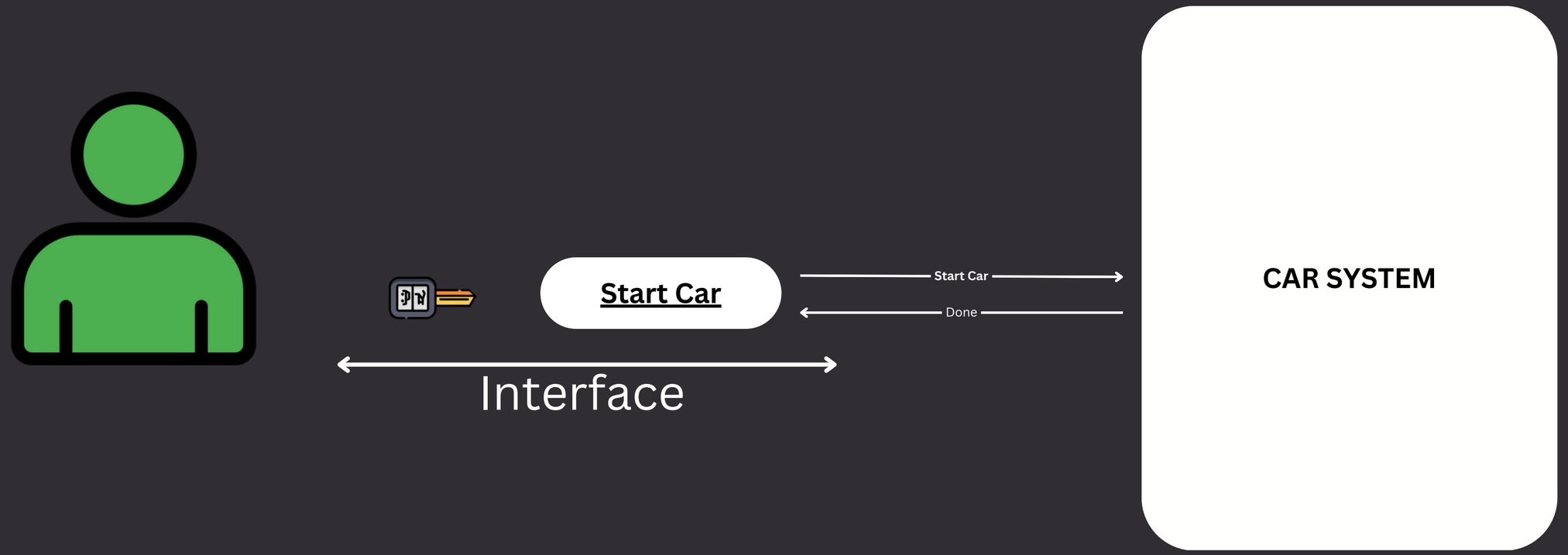
← Done

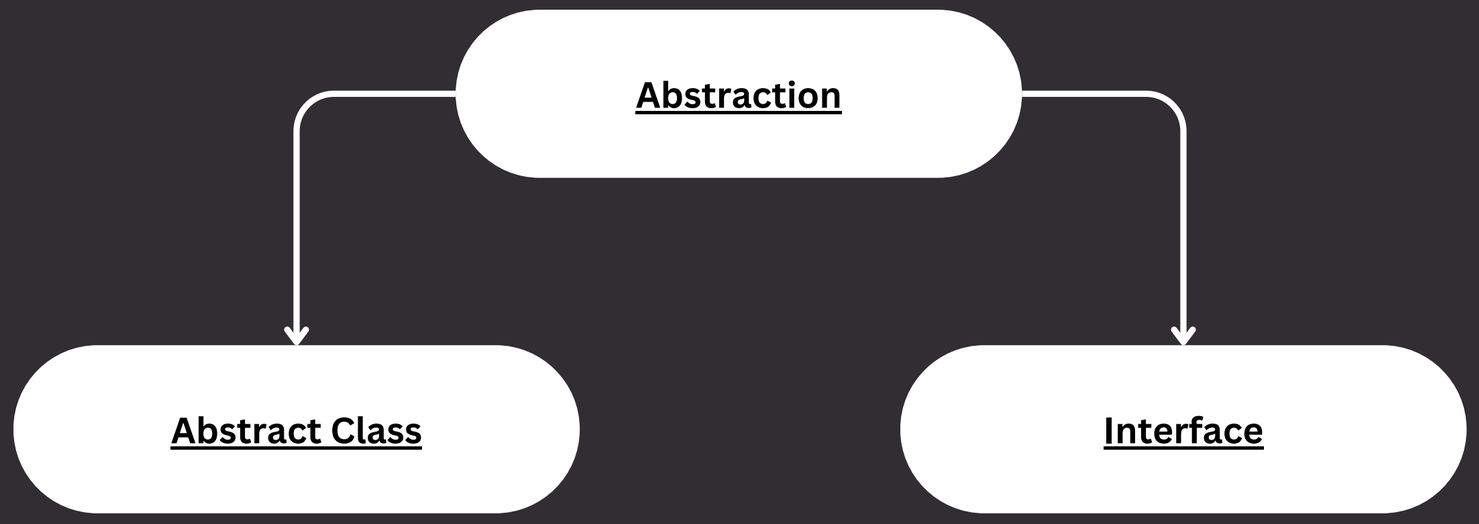
CheckEngine()
CheckElectrical()
CheckFuel()
CheckNecessary()

```java
public abstract class Car {
    protected String brand;
    protected String model;
    protected boolean isEngineRunning;
    protected int fuel;


    public void startCar() {
        if (fuel > 0) {
            checkEngine();
            initializeSystems();
            isEngineRunning = true;
            System.out.println("ماشین روشن شد");
        }
    }

    abstract void checkEngine() {}
    abstract void checkFuel() {}
    abstract void checkElectrical() {}
    abstract void checkNeseccaries() {}

}
```

```java
// پیاده‌سازی برای BMW
public class BMW extends Car {
    public BMW(String model) {
        super("BMW", model);
    }

    @Override
    void checkEngine() {
        System.out.println("بررسی موتور BMW...");
        if (temperature > 90) {
            engineHealthy = false;
            System.out.println("هشدار: دمای موتور بالاست");
        }
    }

    @Override
    void checkFuel() {
        System.out.println("بررسی سیستم سوخت BMW...");
        if (fuel < 10) {
            System.out.println("هشدار: سوخت کم است");
        }
    }

    @Override
    void checkElectrical() {
        System.out.println("بررسی سیستم برق BMW...");
        if (Math.random() < 0.1) { // احتمال مشکل 10%
            electricalSystemOK = false;
            System.out.println("مشکل در سیستم برق");
        }
    }

    @Override
    void checkNeseccaries() {
        System.out.println("بررسی سیستم‌های ضروری BMW...");
    }
```


Start Car
Start Car
Done
CAR SYSTEM


Start Car
Start Car
Done
CAR SYSTEM
Interface


Abstraction
Abstract Class
Interface

When Use ?

Interface

- When you want to define a contract without inheritance
- When you need multiple inheritance
- When you want to specify the behavior of a data type

```java
// تعریف قرارداد برای سیستم‌های پرداخت
public interface PaymentSystem {
    // Behavior Definition - تعریف رفتارها بدون پیاده‌سازی
    void processPayment(double amount);

    // Multiple Implementation - امکان پیاده‌سازی چندگانه
    boolean verifyTransaction(String transactionId);

    // Type Specification - تعریف ثابت‌ها
    public static final double MIN_AMOUNT = 1.0;
    public static final String CURRENCY = "USD";

    // Default method - متد پیش‌فرض (از جاوا 8 به بعد)
    default boolean validateAmount(double amount) {
        return amount >= MIN_AMOUNT;
    }
}
```

```java
// نمونه پیاده‌سازی
class CreditCardPayment implements PaymentSystem {
    @Override
    public void processPayment(double amount) {
        if (validateAmount(amount)) {
            System.out.println("پرداخت با کارت اعتباری به مبلغ " + amount + " " + CURRENCY);
        }
    }

    @Override
    public boolean verifyTransaction(String transactionId) {
        // پیاده‌سازی تایید تراکنش
        return transactionId != null && !transactionId.isEmpty();
    }
}
```

When Use ?

Abstraction Class

- When you need to share code among related classes
- When classes that extend your abstract class have common
- When you want to declare non-public members

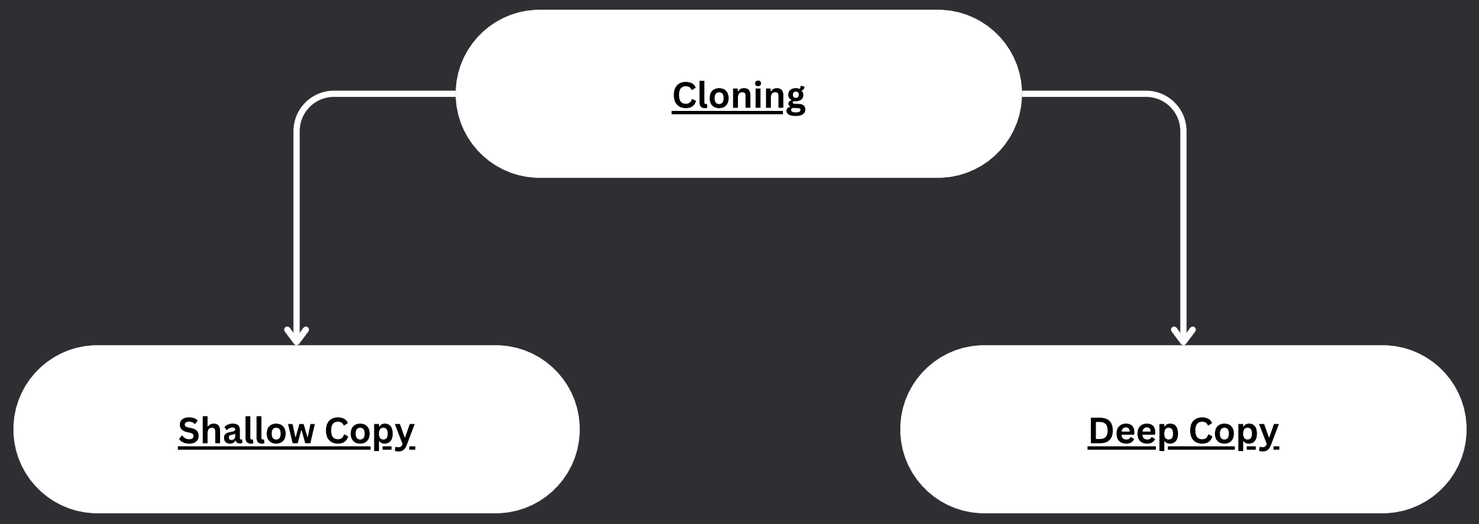
```java
// یک کلاس abstract برای مدیریت وسایل نقلیه
public abstract class Vehicle {
    // State Management - تعریف فیلدها برای مدیریت وضعیت
    protected String brand;
    protected int speed;
    protected boolean isEngineOn;

    // Constructor - سازنده برای مقداردهی اولیه
    public Vehicle(String brand) {
        this.brand = brand;
        this.speed = 0;
        this.isEngineOn = false;
    }

    // Code Sharing - متدهای مشترک با پیاده‌سازی کامل
    public void startEngine() {
        isEngineOn = true;
        System.out.println("موتور روشن شد");
    }

    // Access Control - متد protected برای دسترسی محدود
    protected void checkEngineStatus() {
        if (!isEngineOn) {
            System.out.println("لطفا اول موتور را روشن کنید");
        }
    }

    // Abstract method - متد انتزاعی که باید در کلاس‌های فرزند پیاده‌سازی شود
    public abstract void accelerate(int amount);
}
```

```java
// نمونه پیاده‌سازی
class Car extends Vehicle {
    public Car(String brand) {
        super(brand);
    }

    @Override
    public void accelerate(int amount) {
        checkEngineStatus(); // استفاده از متد protected
        if (isEngineOn) {
            speed += amount;
            System.out.println("سرعت خودرو: " + speed);
        }
    }
}
```

معیار	Abstract Class	Interface
ویژگی‌های مشترک	✅ زمانی که کلاس‌ها دارای فیلدها و متغیرهای مشترک هستند	× نمی‌تواند state داشته باشد
نوع پیاده‌سازی	✅ می‌تواند پیاده‌سازی مشترک داشته باشد	× فقط تعریف متد (تا قبل از جاوا 8)
مدیریت state	✅ می‌تواند state را مدیریت کند	× بدون state
نوع رابطه	✅ "is-a" (مثال: گربه یک حیوان است)	✅ "can-do" (مثال: می‌تواند پرواز کند)
وراثت چندگانه	× فقط یک کلاس را می‌تواند extend کند	✅ می‌تواند چندین interface را implement کند
انعطاف‌پذیری	🟡 انعطاف‌پذیری کمتر	✅ انعطاف‌پذیری بیشتر
پیاده‌سازی	🟡 می‌تواند برخی متدها را پیاده‌سازی کند	✅ کاملاً متفاوت در هر کلاس

Cloning

Object cloning in Java refers to creating an exact copy of an object


Cloning
Shallow Copy
Deep Copy

Shallow Copy

reference to the object is copied

```java
class LargeObject {
    private byte[] largeData = new byte[100_000_000]; // 100MB
    private String name;
    private int id;
}

// مثال کاربردی
public class CacheSystem {
    private Map<String, LargeObject> cache = new HashMap<>();

    public void addToCache(String key, LargeObject obj) throws CloneNotSupportedException {
        // 100 داده MBفقط رفرنس کپی می‌شود، نه کل
        cache.put(key, (LargeObject) obj.clone());
    }
}
```

Usage In Caching

Deep Copy

creating a completely new copy of the object

Why Deep Copy?
REDO

```java
public class Order implements Cloneable {
    private long orderId;
    private Customer customer;
    private List<OrderItem> items;
    private PaymentDetails payment;

    @Override
    public Order clone() throws CloneNotSupportedException {
        Order clonedOrder = (Order) super.clone();

        // کپی عمیق مشتری
        clonedOrder.customer = this.customer.clone();

        // کپی عمیق آیتم‌ها
        clonedOrder.items = new ArrayList<>();
        for (OrderItem item : this.items) {
            clonedOrder.items.add(item.clone());
        }

        // کپی عمیق اطلاعات پرداخت
        clonedOrder.payment = this.payment.clone();

        return clonedOrder;
    }
}
```

```java
// استفاده در سیستم
public class OrderSystem {
    public Order duplicateOrder(Order originalOrder) {
        Order newOrder = originalOrder.clone();
        newOrder.setOrderId(generateNewId());
        return newOrder;
    }
}
```

Inner Class

Type Of Inner Class

1. Nested Inner Class
2. Method Local Inner Classes
3. Static Nested Classes
4. Anonymous Inner Classes

Advantages

- Encapsulation
- readable
- Optimizing the code module.
- EZ Implement Interface

Encapsulation

```java
public class BankAccount {
    private String accountNumber;
    private double balance;
    private String ownerName;
    private String securityPin;
    private List<Transaction> transactionHistory = new ArrayList<>();

    public BankAccount(String accountNumber, String ownerName, String securityPin) {
        this.accountNumber = accountNumber;
        this.ownerName = ownerName;
        this.securityPin = securityPin;
        this.balance = 0.0;
    }


}
```

```java
public class BankAccount {
    private String accountNumber;
    private double balance;
    private String ownerName;
    private String securityPin;
    private List<Transaction> transactionHistory = new ArrayList<>();

    public BankAccount(String accountNumber, String ownerName, String securityPin) {
        this.accountNumber = accountNumber;
        this.ownerName = ownerName;
        this.securityPin = securityPin;
        this.balance = 0.0;
    }


}
```

```java
public class BankAccount {
    ...
        // کلاس درونی معمولی - برای مدیریت تراکنشها
    public class TransactionManager {
        public void deposit(double amount) {
            if (amount > 0) {
                balance += amount; // دسترسی مستقیم به متغیر خصوصی balance
                transactionHistory.add(new Transaction("DEPOSIT", amount));
                System.out.println("Deposit successful. New balance: " + balance);
            }
        }

        public boolean withdraw(double amount, String pin) {
            if (pin.equals(securityPin) && amount > 0 && balance >= amount) { // دسترسی به securityPin
خصوصی
                balance -= amount;
                transactionHistory.add(new Transaction("WITHDRAW", amount));
                System.out.println("Withdrawal successful. New balance: " + balance);
                return true;
            }
            System.out.println("Withdrawal failed.");
            return false;
        }

        public void printStatement() {
            System.out.println("Account Statement for: " + accountNumber + " (" + ownerName + ")");
            System.out.println("Current Balance: " + balance);
            System.out.println("Transaction History:");
            for (Transaction t : transactionHistory) {
                System.out.println(t.getType() + ": " + t.getAmount());
            }
        }
    }

}
```

```java
public class BankAccount {
    ...

        // کلاس درونی استاتیک - برای اعتبارسنجی ورودی‌ها
    public static class AccountValidator {
        public static boolean isValidAccountNumber(String accNumber) {
            return accNumber != null && accNumber.matches("\\d{10}");
        }

        public static boolean isValidPin(String pin) {
            return pin != null && pin.matches("\\d{4}");
        }
    }


}
```

```java
public class BankAccount {
    ...

    public void performAudit(String auditorId) {
        // کلاس درونی محلی - برای بررسی حساب توسط حسابرس
        class Auditor {
            private String id;

            Auditor(String id) {
                this.id = id;
            }

            void auditAccount() {
                System.out.println("Audit performed by auditor " + id);
                System.out.println("Account: " + accountNumber);
                System.out.println("Owner: " + ownerName);
                System.out.println("Balance: " + balance);
                System.out.println("Transaction count: " + transactionHistory.size());
            }
        }

        Auditor auditor = new Auditor(auditorId);
        auditor.auditAccount();
    }

}
```

```java
public class BankAccount {
    ...

    public void addInterestNotification() {
        // کلاس درونی بی‌نام - برای اعلان‌ها
        Runnable notification = new Runnable() {
            @Override
            public void run() {
                double interest = balance * 0.05; // دسترسی به balance
                System.out.println("Dear " + ownerName + ", an interest of " +
                                   interest + " has been added to your account.");
            }
        };

        notification.run();
    }

    // کلاس درونی برای نگهداری اطلاعات تراکنش
    private class Transaction {
        private String type;
        private double amount;
        private LocalDateTime timestamp;

        Transaction(String type, double amount) {
            this.type = type;
            this.amount = amount;
            this.timestamp = LocalDateTime.now();
        }

        public String getType() { return type; }
        public double getAmount() { return amount; }
    }
}
```

Q&A

Thank You

Amirmahdi Mashayekhi

telegram : @mashmool0

telegram Channel: @mashmoolcode