

# Interfaces and Polymorphism

---



**Dan Wahlin**

WAHLIN CONSULTING

@danwahlin [www.codewithdan.com](http://www.codewithdan.com)



# Module Overview

The Role of Interfaces

Creating an Interface

Using Interfaces

Interfaces, Classes, and  
Polymorphism



# The Role of Interfaces

---



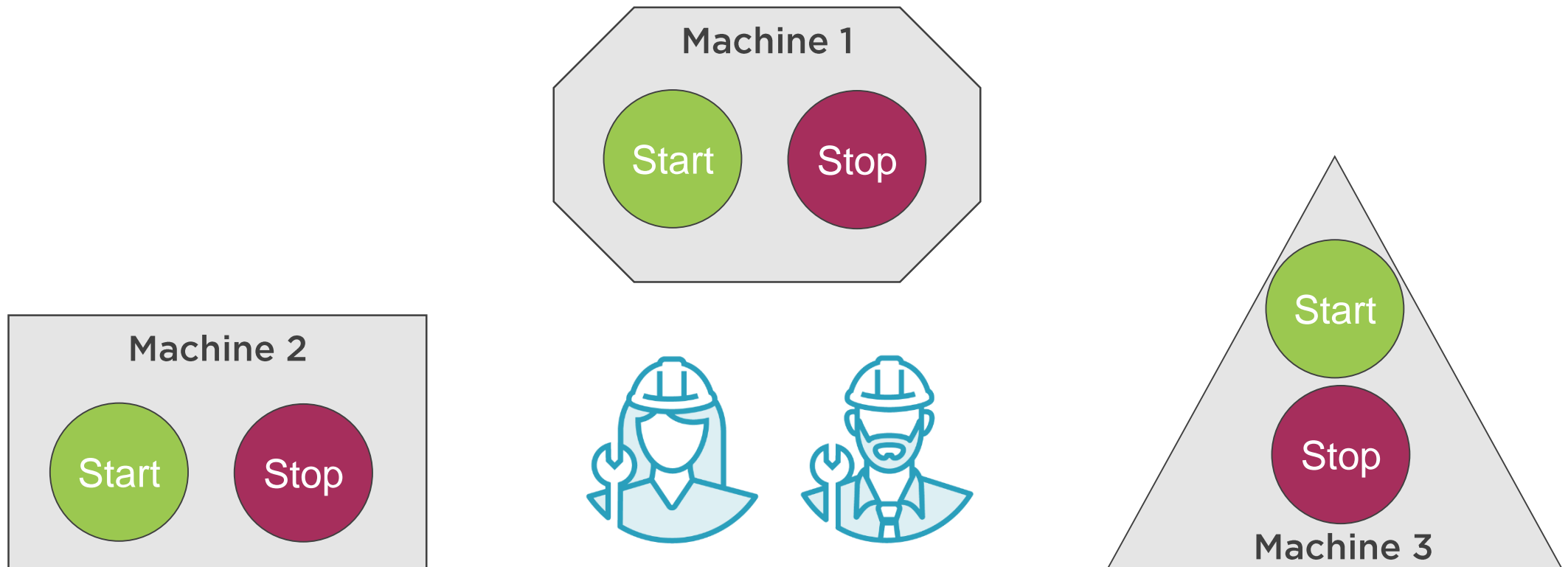
# Interfaces

Interfaces act as a contract that defines a set of rules.



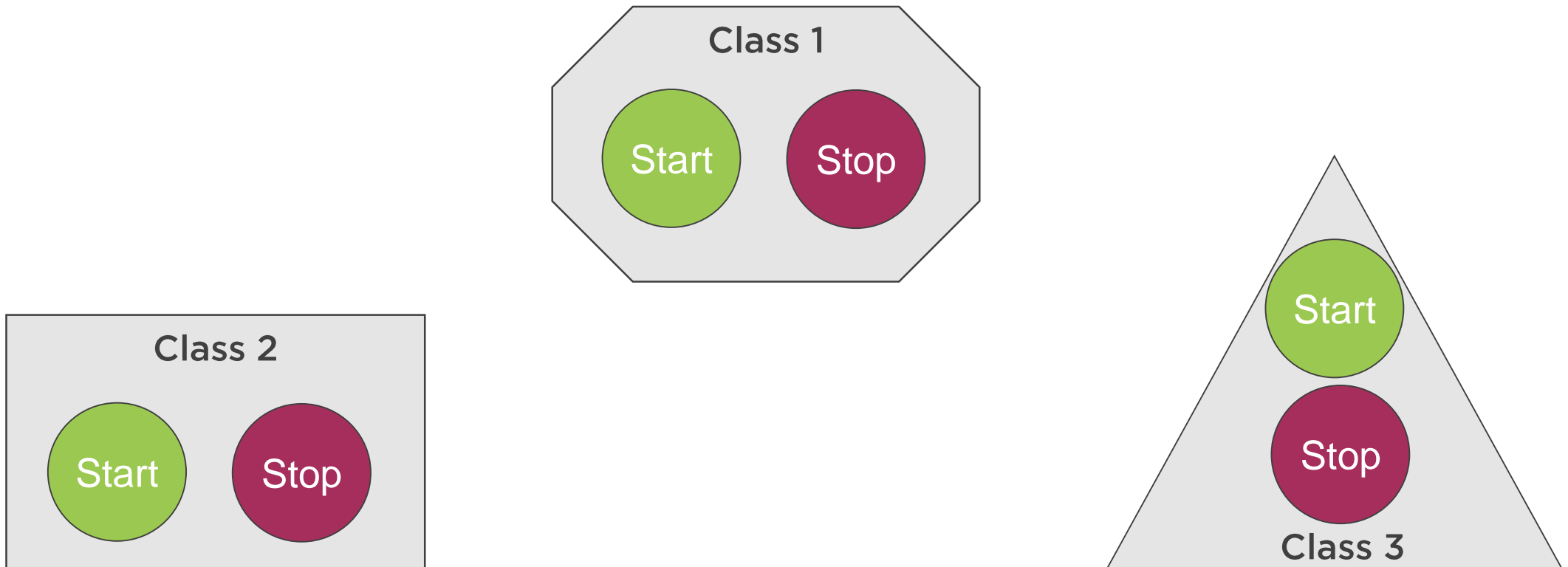
# What Is an Interface?

Workers may require all machines to have a standard "interface" to simplify training and usage

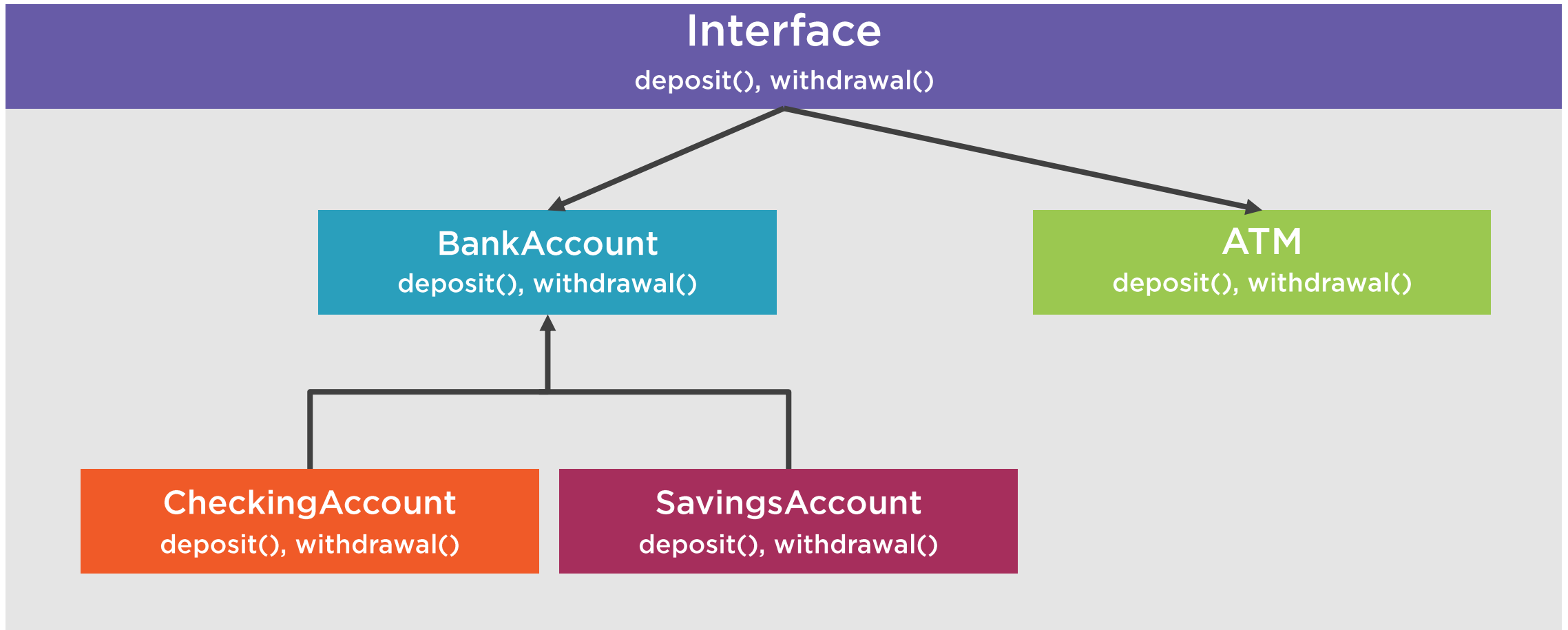


# Enforcing Consistency Across Classes

Interfaces can be used to enforce consistency across different classes in an application



# The Role of Interfaces



```
class BankingAccount {  
    getAccountInfo() {  
        return {  
            routingNumber: 845269787,  
            bankNumber: 123538954  
        };  
    }  
}
```

## What Is This Method Returning?

**Interfaces** can be used to help define the "shape" of data being returned from a method





```
class MyObject {  
    private _settings: any;  
    constructor(settings: any) {  
        this._settings = settings;  
    }  
}
```

## What Should You Pass to This Constructor?

Interfaces can be used to help identify the type of data to pass to a constructor or method



# Class or Interface?



**0 Calories - No fat!**



# Creating an Interface

---



```
interface DepositWithdrawal {  
    deposit(amount: number): void;  
    withdrawal(amount: number): void;  
}
```

```
interface AccountInfo {  
    routingNumber: number;  
    bankNumber: number;  
}
```

## Creating an Interface

Interfaces can be used to create custom data types

Classes can implement interfaces which can help ensure consistency across an application

Interfaces are only used during development (no impact on script/bundle size)

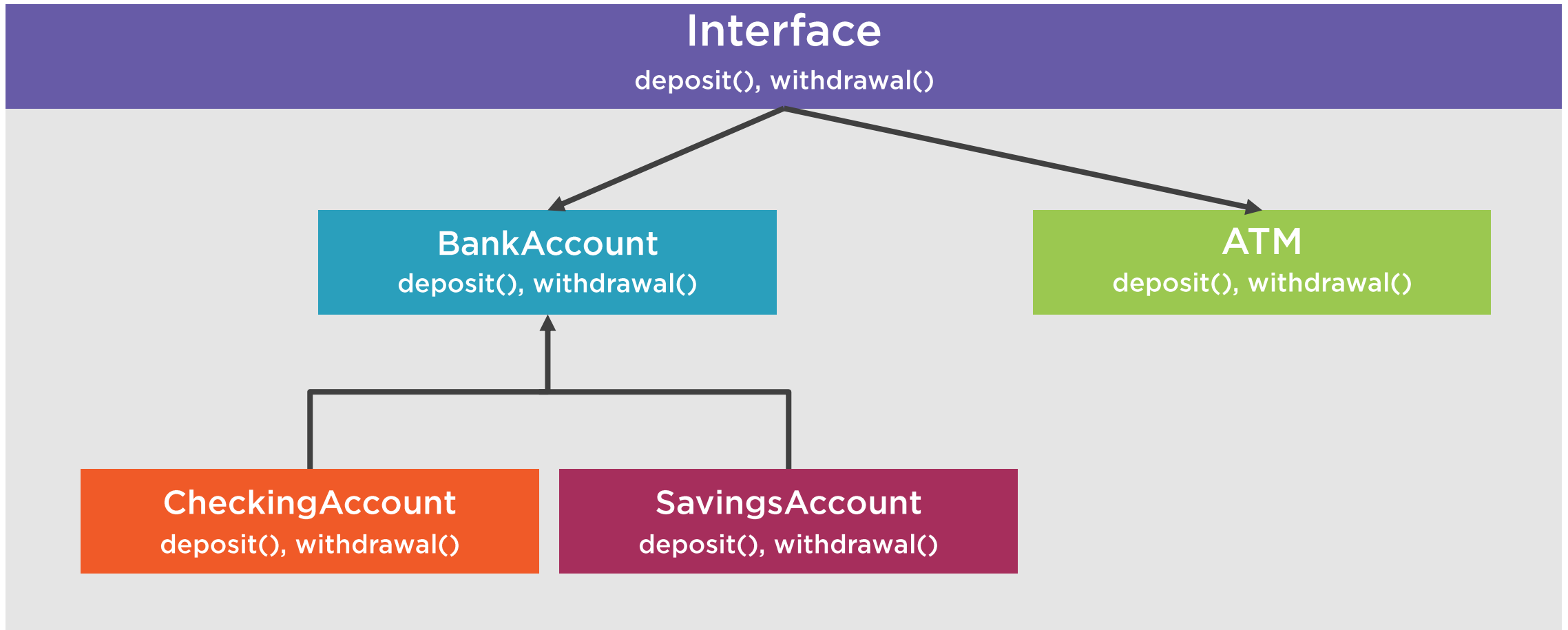



# Using Interfaces

---



# Implementing an Interface





```
class ATM implements DepositWithdrawal {  
    deposit(amount: number) { ... }  
    withdrawal(amount: number) { ... }  
}  
  
interface DepositWithdrawal {  
    deposit(amount: number): void;  
    withdrawal(amount: number): void;  
}
```

## Implementing an Interface

Classes can implement interfaces using the TypeScript **implements** keyword



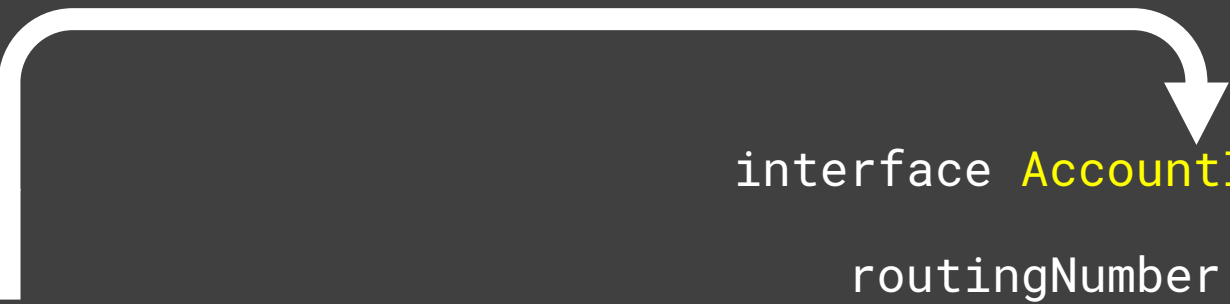


# Polymorphism

Objects exhibit the same behavior but in a different way.



```
accountInfo: AccountInfo;
```



```
interface AccountInfo {  
    routingNumber: number;  
    bankNumber: number;  
}
```

## Using Interfaces as Types

Interfaces can be used as types to define the "shape" of data held in a property or that is passed to a function/method or constructor



# Interfaces, Classes, and Polymorphism

---



# Summary



Interfaces are code contracts

Key benefits of using interfaces include:

- Drive consistency across multiple objects
- Define the "shape" of data passed to a constructor or function/method
- Use as a custom data type

Classes can implement an interface by using the **implements** keyword

Abstract classes and interfaces can both be used to achieve polymorphic behavior

