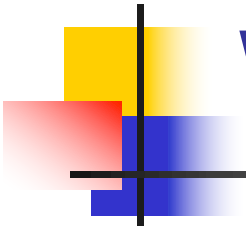
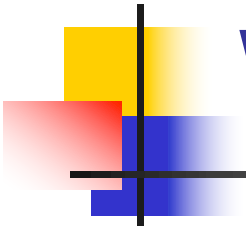
The Visitor Design Pattern



What's It All About?

- Allows for new operations to be defined and used on elements of an object structure with out changing the contents of those elements.
- The Key is Double Dispatch

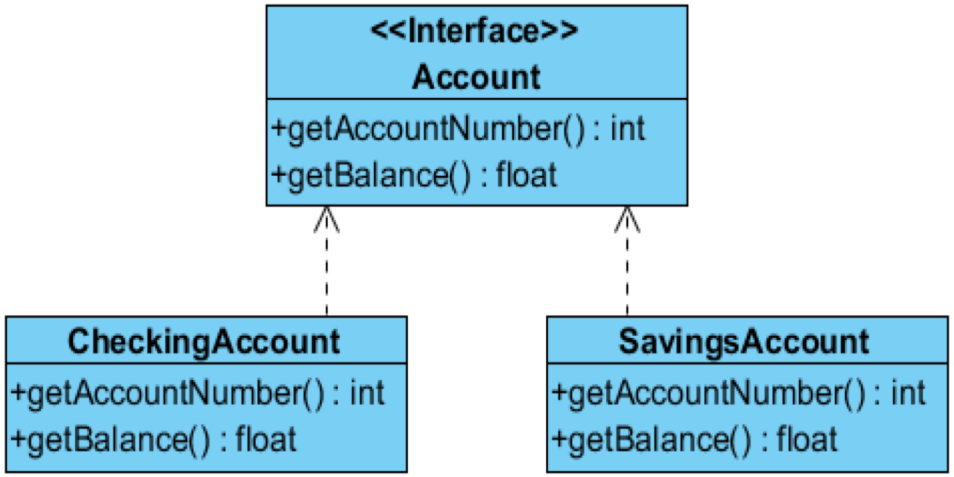


Where Applicable

- Rarely Changing Object Structures
- Using Unrelated Operations
- Many Classes with Differing Interfaces

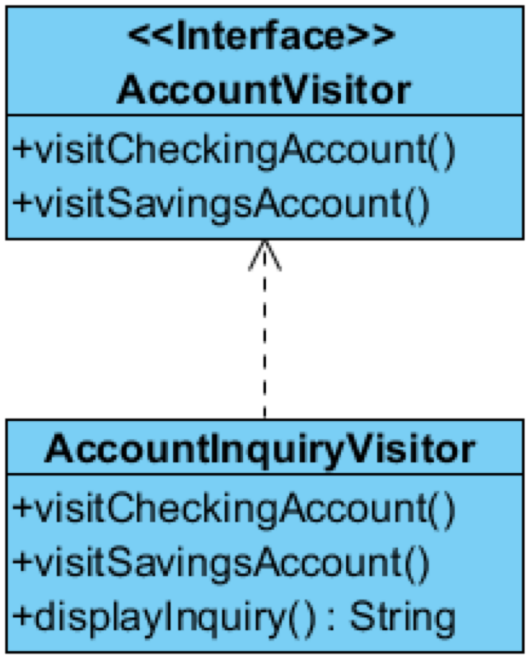
How it Works

- Concrete Object Structure
- Assume Rarely Changing
- Bank Accounts



Add an Inquiry Operation

- Inquiry to Display Accounts
- Don't Want to Change Structure
- Create a Visitor Structure
- Account Visitor



Account Visitor Interface

```
public interface AccountVisitor {
    public void visitCheckingAccount(CheckingAccount cAccount);
    public void visitSavingsAccount(SavingsAccount sAccount);
}
```

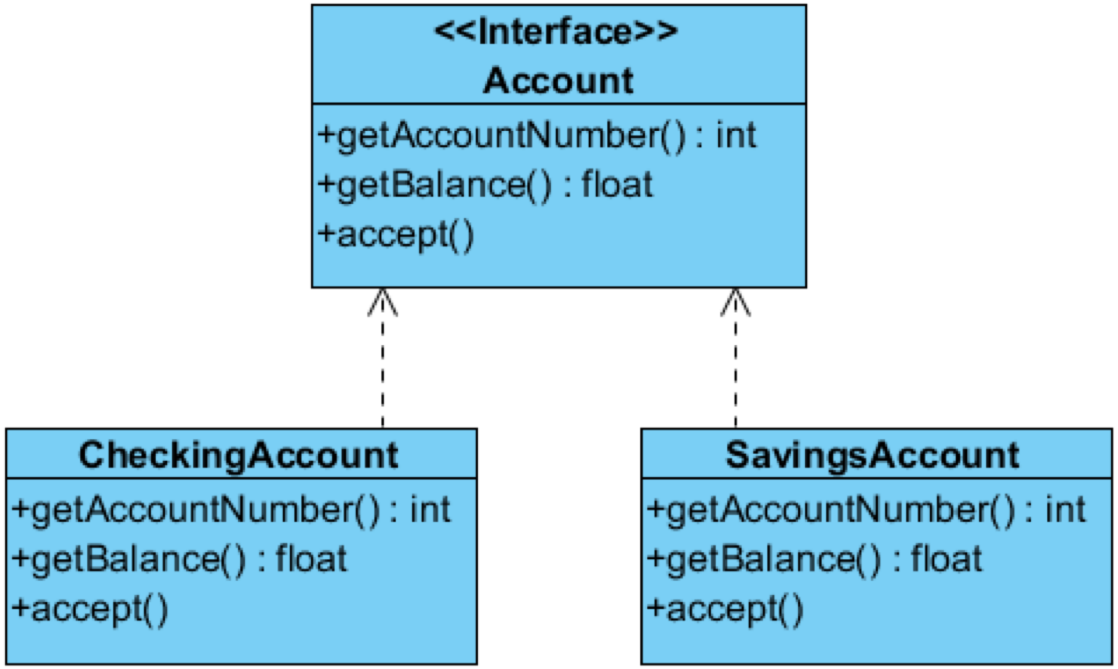
Inquiry Visitor

```
import java.text.DecimalFormat;

public class AccountInquiryVisitor implements AccountVisitor {
    private String checkAcctBal;
    private int checkAcctNum;
    private String saveAcctBal;
    private int saveAcctNum;
    private DecimalFormat money;

    public AccountInquiryVisitor() {
        checkAcctBal = "";
        checkAcctNum = 0;
        saveAcctBal = "";
        saveAcctNum = 0;
        money = new DecimalFormat("$0.00");
    }
    public void visitCheckingAccount(CheckingAccount cAccount) {
        checkAcctBal = money.format(cAccount.getBalance());
        checkAcctNum = cAccount.getAccountNumber();
    }
    public void visitSavingsAccount(SavingsAccount sAccount) {
        saveAcctBal = money.format(sAccount.getBalance());
        saveAcctNum = sAccount.getAccountNumber();
    }
    public String displayInquiry() {
        String inquiry = "";
        inquiry = inquiry + "Balance Inquiry for All Accounts:\n";
        inquiry = inquiry + "\n";
        inquiry = inquiry + "  Checking " + checkAcctNum + "\n";
        inquiry = inquiry + "  Current Balance: " + checkAcctBal + "\n";
```

Account Structure Change



Account Interface

```
public interface Account {
    public int getAccountNumber();
    public float getBalance();
    public void accept(AccountVisitor visitor);
}
```

Checking Account

```
public class CheckingAccount implements Account {
    private int accountNumber;
    private float currentBalance;

    public CheckingAccount(int accountNumber, float initialAmount) {
        this.accountNumber = accountNumber;
        currentBalance = initialAmount;
    }
    public int getAccountNumber() {
        return this.accountNumber;
    }
    public float getBalance() {
        return currentBalance;
    }
    public void accept(AccountVisitor visitor) {
        visitor.visitCheckingAccount(this);
    }
}
```

Savings Account

```
public class SavingsAccount implements Account {
    private int accountNumber;
    private float currentBalance;

    public SavingsAccount(int accountNumber, float initialAmount) {
        this.accountNumber = accountNumber;
        currentBalance = initialAmount;
    }
    public int getAccountNumber() {
        return this.accountNumber;
    }
    public float getBalance() {
        return currentBalance;
    }
    public void accept(AccountVisitor visitor) {
        visitor.visitSavingsAccount(this);
    }
}
```

Main Method

```
public class Main {
    public static void main(String[] args) {
        CheckingAccount myChecking = new CheckingAccount(10253, 1000);

        SavingsAccount mySavings = new SavingsAccount(10334, 2000);

        AccountInquiryVisitor inquiry = new AccountInquiryVisitor();

        myChecking.accept(inquiry);

        mySavings.accept(inquiry);

        System.out.println(inquiry.displayInquiry());
    }
}
```