JavaScript Patterns
7.4, 7.5, 7.6

かんだ　ゆうすけ

7.4 Decorator Pattern

- 実行時に動的に機能をオブジェクトに追加する

```
var sale = new Sale(100); // 価格は100ドル

sale.getPrice(); //100
sale = sale.decorate('fedtax'); //連邦税を追加

sale = sale.decorate('quebec'); //地方税を追加

sale = sale.decorate('money');  //金額を書式化

sale.getPrice(); // "$112.88"
```

- Javaでよく見るコレ

```
//バッファリングしつつファイルを読み込む(in Java)
BufferedReader br = new BufferedReader(new FileReader(file));
```

7.4.2 Decorator Pattern実装1

- 機能追加ごとにオブジェクトを継承、
  スーパークラス(クラスじゃないけど)の同じメソッドを
  呼び出す

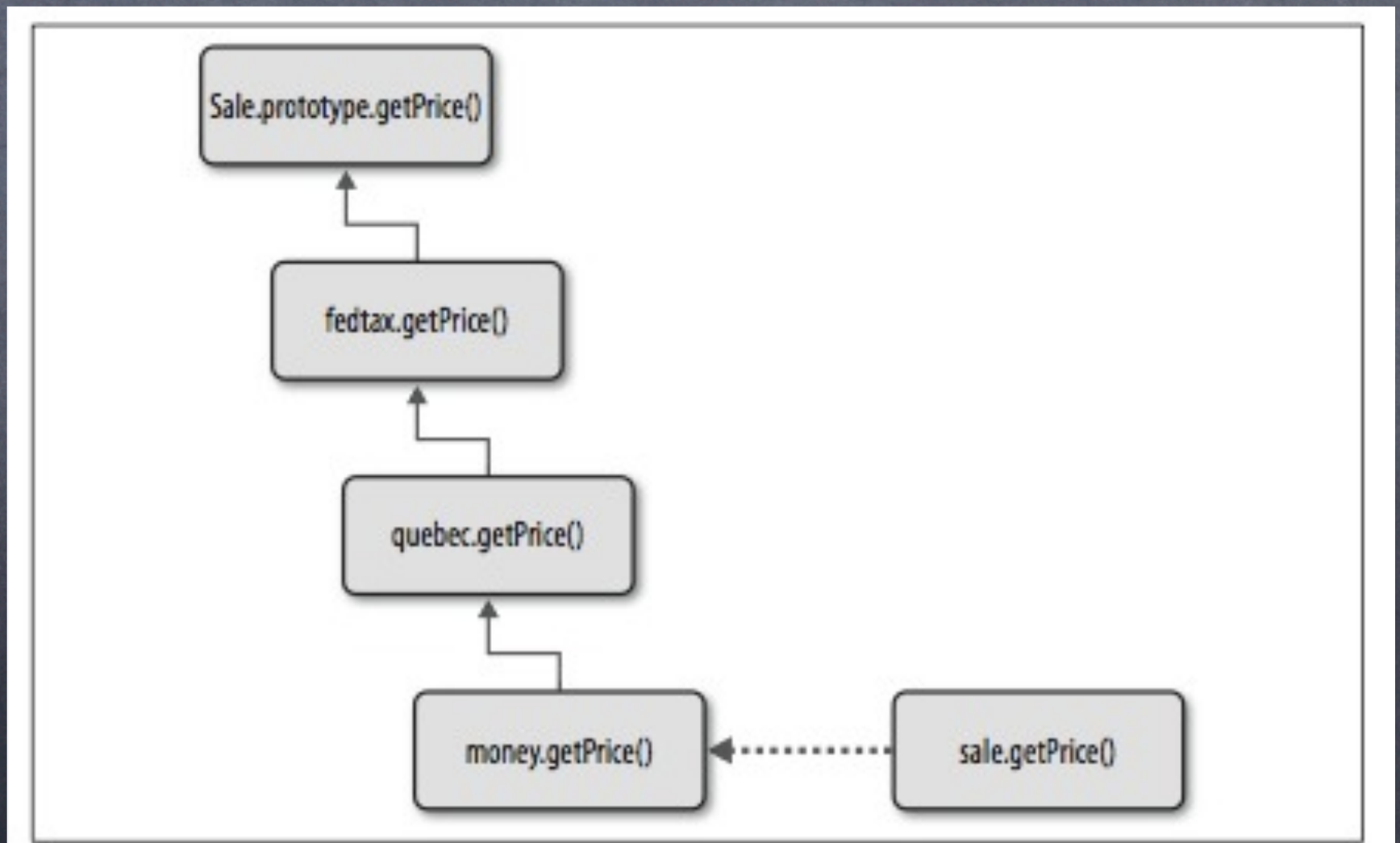

図7-1　デコレータパターンの実装

7.4.3 Decorator Pattern実装2

- 修飾する項目をオブジェクト内の配列で保持
- 修飾されるべきメソッド呼び出しの際に
  上記配列を捜査し、Decoratorを呼び出す

7.5 Strategy Pattern

- 実行時にアルゴリズムを選択するための
  デザインパターン

```
var data = { //このデータをバリデーションすることを考える
    first_name: "Super",
    last_name: "Man",
    age: "unknown",
    username: "o_O"
};

validator.config = {  //バリデーションのStrategy
    first_name: 'isNonEmpty',
    age: 'isNumber',
    username: 'isAlphaNum'
};
validator.validate(data); //validator.configに基づくバリデーションを実行

if (validator.hasErrors()) {
    print(validator.messages.join("\n"));
}
```

7.6 Facade Pattern

- オブジェクトに代替のインタフェースを提供する
  よく使う処理の組み合わせを集約する

//stopPropagation()
//イベントを停止して、親ノードに伝搬させない。

//preventDefault()
//ブラウザにデフォルトの動作をさせない(例：リンクやフォームの送信を無効にする)。

```
var myevent = { // ...
  stop: function (e) {
    e.preventDefault();
    e.stopPropagation();
  }
// ... };
```