Accessorを使用するとJavaコードの堅牢性が高まります

Scott W. Ambler

2000年 10月 12日

Scott Amblerがaccessorの使用方法を説明し、そしてなぜaccessorを開発標準の１つとして推進していくべきかを述べています。この記事は、The Object Primer 2nd Edition の第8章を変更したものです。

カプセル化と情報の隠蔽は、堅固なクラスおよびコンポーネントを設計するための基本原則です。カプセル化を増強したり情報の開示を防ぐために、Java開発者が利用できる大切なテクニックとして、accessorメソッドを適切に使用するということが挙げられます。

accessor -- フィールドの値を直接操作するメンバー関数 -- には、setterとgetterの2つのフレーバーがあります。setterはフィールドの値を変更し、getterはフィールドの値を取得します。accessorは最小限のオーバーヘッドをコードに与えますが、パフォーマンスの劣化は、多くの場合、他の要因 (問題のあるデータベース設計など) に比べて取るに足らないものです。accessorはクラスのインプリメンテーション詳細を隠すのに役立ちますので、コードの堅牢性を高めます。フィールドへのアクセスが行われる制御ポイントを最大2つ (1つのsetterと1つのgetter) 用意することにより、変更が必要な個所を最低限に抑えられ、クラスの保守容易性を高めることができます。

図1 は、`Person` クラスの`homeAddress` フィールドに対するgetterおよびsetterメンバー関数のインプリメンテーションを表しています。getterおよびsetterは、データ・フィールドにあてはまる重要なビジネス・ルール、トランスフォーメーション・ロジック、および検証ロジックにカプセル化を適用することができるので役に立ちます。たとえば、図2 はsetterメンバー関数の代替インプリメンテーションを示しています。渡されたaddressオブジェクトが有効であることを`validate()` メンバー関数を呼び出して最初に妥当性検査をする仕組みを見てください。有効であれば、この関数は`homeAddress` フィールドの値を設定します。無効であれば、関数は例外をthrowします。

図1.`Person` クラスの`homeAddress` フィールドに対するgetterおよびsetter

```
/**
* Answers the person's home address
*
* @return homeAddress
*/
public Address getHomeAddress()
{
  return homeAddress;
}
/**
* Sets the person's home address
*
* @param homeAddress
*/
public void setHomeAddress(Address homeAddress)
{
  this.homeAddress = homeAddress;
}
```

図2.`setHomeAddress()` メンバー関数の代替インプリメンテーション

```
/**
* Sets the person's home address
*
* @param homeAddress
* @return homeAddress
*/
public Address setHomeAddress(Address homeAddress) throws InvalidDataException
{
  // Only set the address if it is valid
  if (homeAddress.validate() ) {
        this.homeAddress = homeAddress;
  }
  else {
        throw new InvalidDataException();
  }
}
```

組織の中で強化できる重要な標準の1つに、accessorの使用があります。開発者の中には、必要とされる若干の追加キーストローク (たとえばgetterの場合、フィールド名に加えてget と () を入力する必要があります) が面倒であるという理由で、accessorメンバー関数の使用を好まない人もいます。しかし、accessorを使用することによって強化される保守容易性と拡張性は、使用する際の手間を補って余りあるものです。

getterメンバー関数には、get +フィールド名 という形式の名前を指定する必要があります。ただし、フィールドがブール (真または偽) を表す場合には、getterにはis +フィールド名 という形式の名前を指定します。setterメンバー関数の場合には、フィールド・タイプにかかわらず、set +フィールド名 という形式の名前を指定する必要があります。表1 で示すように、フィールド名は、常に 各ワードの先頭文字を大文字にして、大文字小文字混在で表します。この命名規則は、Java Development Kit (JDK) では一貫して使用され、JavaBeans開発およびEnterprise JavaBean (EJB) 2.0のパーシスタント・フィールドでは必須です。

表1. accessor名の例

フィールド	タイプ	getter名	setter名

name	ストリング	getName	setName
homeAddress	Address オブジェクト	getHomeAddress	setHomeAddress
persistent	ブール	isPersistent	setPersistent
zipCode	int	getZipCode	setZipCode
instructors	Professor オブジェクトのベクトル	getInstructors	setInstructors

関連トピック

- The Java Language Specification、James Gosling、Bill Joy、Guy Steele共著。Reading, MA: Addison-Wesley Longman, Inc., 1996.
- Advanced Java: Idioms, Pitfalls, Styles and Programming Tips、Chris Laffra著。Upper Saddle River, NJ: Prentice Hall Inc., 1997.
- Essential Java Style: Patterns for Implementation、Jeff Langr著。Upper Saddle River, NJ: Prentice-Hall PTR, 1999.
- Java 2 Performance and Idiom Guide: Guidelines for Java 2 Performance, Coding, and Testing 、Craig Larman、Rhett Guthrie共著。Upper Saddle River, NJ: Prentice Hall Inc., 2000.
- Object-Oriented Software Construction, Second Edition、Bertrand Meyer著。Upper Saddle River, NJ: Prentice-Hall PTR, 1997.
- The Elements of Java Style、Alan Vermeulen、Scott W. Ambler、Greg Bumgardner、Eldon Metz、Trevor Misfeldt、Jim Shur、Patrick Thompson共著。New York: Cambridge University Press, 2000.