# **授業のポイント**

1. Iteratorパターンについて学習します。

# **学習項目**

・Iteratorパターン

# **J2Kad25D「ECCフーズ（ループの復習）」**

# **１．内部仕様の異なるデータ**

ECCフーズが買収したECCコーヒーとECCドーナツではメニュー管理方法が異なっているため、メニュー表示処理もそれぞれの店用で作成する必要があります。グループが増えていくと、それぞれの店用に表示処理を作成する必要があり、mainメソッドにとって非常に面倒です。

# **J2Kad25C「ECCフーズ（ループの本質）」**

# **１．ループ構造の本質**

データを順番に処理していくループは「次のデータが存在するのか？（hasNext）」と「データを取得する（next）」の2つのメソッドがあれば、構成することができます。

while( データがある？ ) {

データを取得して処理をする

}

このhasNextとnextに対応したオブジェクトをイテレータと呼びます。CafeMenu、DonutMenuそれぞれに対応するイテレータ（CafeIterator、DonutIterator）を作成すれば、同じ形のループにすることができます。

# **２．MenuIteratorによるループの共通化**

ループ構造が同じになってもイテレータの種類が異なっていると、コードをひとつにまとめることができません。そこでMenuIteratorインターフェイスを導入して、CafeIteratorとDonutIteratorに実装します。これでループ処理のコードを共通化できます。

1. CafeMenu（またはDonutMenu）を生成
2. CafeIterator（またはDonutIterator）に渡す情報を取得
3. CafeIteretor（またはDonutIterator）を生成
4. 表示（共通処理）

①～③はCafeMenuとDonutMenuによって仕様が異なりますが、④の表示処理は同じコードで処理できます。

# **J2Kad25B「ECCフーズ（Iteratorパターン）」**

# **１．さらなる簡略化**

CafeIteratorやDonutIteratorを生成するためには、対応する店（CafeMenu、DonutMenu）からメニューに関する情報を取得してコンストラクタに渡す必要があります。このイテレータの生成処理をCafeMenuやDonutMenuに任せるようにすれば、mainはどんな情報が必要かを知らなくてもイテレータを生成できます。さらにイテレータ生成も共通インターフェイス（Menuインターフェイス）にすれば、この部分の処理も共通化できます。そもそもCafeIteratorに渡す情報はCafeMenuが知っているので、CafeMenuに生成させるというのは理にかなっています（DonutIteratorの生成も同じ）。

# **２．Iteratorパターン**

Iteratorパターンとは、コンテナオブジェクトの要素を列挙する手段を独立させることによってコンテナの内部仕様に依存しない反復子（イテレータ）を提供するパターン（Wikipediaより）です。ここではIteratorパターンを使うことで、メニュー表示の簡略化を行っています（ただし最終形はJ2Kad25S）。

# **J2Kad25A「ECCフーズ（M&A）」**

# **１．ECCバーガーの買収**

ECCバーガー（BurgerMenuクラス）のメニュー表示を追加します。J2Kad25Bまでの復習になりますが、ArrayListを使っている点に注意が必要です。要素数（sizeメソッド）や要素の取得（getメソッド）など、ArrayListの使い方を思い出す必要があります。

# **J2Kad25S「ECCフーズ（匿名クラス）」**

# **１．内部クラス（実践編P.125「内部クラスとは」）**

main（使う側）はMenuインターフェイスとMenuIteratorインターフェイスの仕様さえ知っていれば、CafeMenuもDonutMenuもBurgerMenuも表示することができます。各イテレータクラスの具体的な仕様は対応するメニュークラスが知っていればOKです。よって各イテレータクラスは対応するメニュークラスの内部クラスにすることができます。

内部クラスにすることによって、イテレータクラスはメニュークラスのprivateメンバを直接参照することができるようになります。よって各メニュークラスの情報への参照やコンストラクタによる設定が不要になります。

# **２．匿名クラス（実践編P.127「匿名クラス」）**

さらにイテレータクラスはクラス名さえ必要ありません。よってMenuIteratorインターフェイスの匿名クラスにすることができます。

# **J2Kad25X「ArrayListのイテレータ」（実践編P.108「イテレータ」）**

# **１．コレクションのイテレータ**

ArrayListなどのコレクションにもイテレータの機能が備わっています（使い方はMenuIteratorとほぼ同じです）。

Iterator<MenuItem> it = menu.iterator(); // イテレータの取得

while(it.hasNext()) {

// it.next()でMenuItemを取得

}

# **２．オブジェクトアダプター（委譲を使ったアダプター）※次回使う予定**

MenuIteratorインターフェイスを通してArrayListのイテレータを使用します。匿名クラスにArrayListのイテレータを生成し、MenuIteratorのhasNext・nextでこのイテレータを操作（委譲）します。簡単なオブジェクトアダプター（インスタンスへの委譲を使ったAdapterパターン）です。