● 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。