# **世界にはばたくECCフーズ！**

世界にはばたくECCフーズは外食チェーン店を次々と買収している。現在傘下にあるのは、ECCコーヒーとECCドーナツ、さらに他のチェーン店の買収も計画している。買収後は、すべての店でお互いのメニューの導入を進める予定だ。そこで全チェーン店のメニューを表示するシステムを作成しようとしたところ、それぞれのチェーン店でメニュー管理がバラバラだった（**リスト1、2、3**）。今後の買収（メニュー表示の追加）を考えて、簡略化された処理を作成せよ。

**リスト1：MenuItemクラス（ファイル「Menu.java」）**

メニュー名と値段を格納するクラス。printメソッドで表示する。

class MenuItem {

private String name; // メニュー名

private int price; // 値段

public MenuItem(String name, int price) {

this.name = name;

this.price = price;

}

public void print() { System.out.println(name + "：" + price + "円"); }

}

**リスト2：ECCコーヒーのメニュー（ファイル「CafeMenu.java」）**

MenuItemの配列で管理している。配列の値がnullのとき、それ以上メニューはない。

public class CafeMenu {

private MenuItem[] menu = new MenuItem[100];

public CafeMenu() {

menu[0] = new MenuItem("ドリップコーヒー", 390);

menu[1] = new MenuItem("アールグレイ", 430);

menu[2] = new MenuItem("オレンジジュース", 220);

menu[3] = null; // 終了コード

}

public MenuItem[] getMenu() { return menu; } // menu配列のゲッター

}

**リスト3：ECCドーナツのメニュー（ファイル「DonutMenu.java」）**

メニュー名と値段をそれぞれ別々の配列で管理している。値段の値が-1のとき、それ以上メニューはない。

なぜ、こんな方法で管理しているのか知る由もない（買収したときすでにこうなっていた）。

public class DonutMenu {

private String[] names = new String[100]; // メニュー名

private int[] prices = new int[100]; // 値段

public DonutMenu() {

names[0] = "ハニーディップ"; prices[0] = 120;

names[1] = "ハニーチュロ"; prices[1] = 130;

names[2] = "チョコリング"; prices[2] = 140;

prices[3] = -1; // 終了コード

}

public String[] getNames() { return names; } // メニュー名のゲッター

public int[] getPrices() { return prices; } // 値段のゲッター

}

# **J2Kad25D「ECCフーズ（ループの復習）」**

for文を使ってECCコーヒーとECCドーナツのすべてのメニューを表示する処理を作成せよ。

**リスト1：メニュー表示処理（ファイル「J2Kad25D.java」）**

public class J2Kad25D {

public static void main(String[] args) {

…

while (true) {

System.out.print("どのメニューを表示しますか？（0：ECCコーヒー、1：ECCドーナツ、-1：終了）＞");

int shop = Integer.parseInt(in.next());

if (shop < 0) break;

switch(shop) {

default:

case 0: // ECCコーヒー

**ECCコーヒーの全メニューを表示する**

break;

case 1: // ECCドーナツ

**ECCドーナツの全メニューを表示する**

break;

}

System.out.println();

}

}

}

**課題完成時の画面**

世界にはばたくECCフーズ！

ただいまM&Aで拡大中！！

どのメニューを表示しますか？（0：ECCコーヒー、1：ECCドーナツ、-1：終了）＞**0**

ドリップコーヒー：390円

アールグレイ：430円

オレンジジュース：220円

どのメニューを表示しますか？（0：ECCコーヒー、1：ECCドーナツ、-1：終了）＞**1**

ハニーディップ：120円

ハニーチュロ：130円

チョコリング：140円

どのメニューを表示しますか？（0：ECCコーヒー、1：ECCドーナツ、-1：終了）＞**-1**

# **J2Kad25C「ECCフーズ（ループの本質）」**

ループ構造は「データがあるかどうか（hasNext）」と「データを取得する（next）」で構成することができる。

while( データがある？ ) {

データを取得して処理をする

}

1. CafeIteratorクラスとDonutIteratorクラスを作成し、メニュー表示のループ処理を修正せよ。
2. ファイル「Menu.java」にMenuIteratorインターフェイスを作成し、メニュー表示処理を共通化せよ。

**CafeIteratorクラス（ファイル「CafeMenu.java」に作成）**

|  |  |
| --- | --- |
| メンバ | 説明 |
| private MenuItem[] menu = null; | Caféメニュー配列への参照。コンストラクタで設定。 |
| private int i = 0; | 配列のインデックス。初期値は0。 |
| public CafeIterator(MenuItem[] menu) | コンストラクタ。menuへの参照を設定する。 |
| public boolean hasNext() | i番目のメニューがあればtrue、なければfalseを返す。 |
| public MenuItem next() | i番目のメニューを返し、iの値を1増やす。 |

**DonutIteratorクラス（ファイル「DonutMenu.java」に作成）**

|  |  |
| --- | --- |
| メンバ | 説明 |
| private String[] names = null; | Donutのメニュー名配列への参照。コンストラクタで設定。 |
| private int[] pricesi = null; | Donutの値段配列への参照。コンストラクタで設定。 |
| private int i = 0; | 配列のインデックス。初期値は0。 |
| public DonutIterator(String[] names, int[] prices) | コンストラクタ。namesとpricesへの参照を設定する。 |
| public boolean hasNext() | i番目のメニューがあればtrue、なければfalseを返す。 |
| public MenuItem next() | i番目のメニューをMenuItemとして返し、iの値を1増やす。 |

**課題完成時のクラス図**

|  |
| --- |
| <<interface>>  ***MenuIterator*** |
|  |
| *hasNext() : boolean*  *next() : MenuItem* |

|  |
| --- |
| **CafeIterator** |
| … |
| … |

|  |
| --- |
| **DonutIterator** |
| … |
| … |

**課題完成時の画面**

（J2Kad25Dと同じ）

# **J2Kad25B「ECCフーズ（Iteratorパターン）」**

ファイル「Menu.java」にMenuインターフェイスを作成し、CafeIteratorとDonutIteratorの取得処理を共通化せよ。

**CafeMenuとDonutMenuに追加するメソッド**

|  |  |
| --- | --- |
| メソッド | 説明 |
| public MenuIterator iterator() | CafeMenu：CafeIteratorを生成して返す。  DonutMenu：DonutIteratorを生成して返す。 |

**課題完成時のクラス図**

|  |
| --- |
| <<interface>>  ***Menu*** |
|  |
| *iterator() : MenuIterator* |

|  |
| --- |
| **CafeMenu** |
| … |
| … |

|  |
| --- |
| **DonutMenu** |
| … |
| … |

**課題完成時の画面**

（J2Kad25Dと同じ）

# **J2Kad25A「ECCフーズ（M&A）」**

世界にはばたくECCフーズが今度はあのハンバーガーチェーンを買収した！ハンバーガーチェーンのメニュー表示を追加せよ。

**リスト1：BurgerMenuクラス（ファイル「BurgerMenu.java」）**

ArrayListで管理している。

public class BurgerMenu {

private ArrayList<MenuItem> menu = new ArrayList<>();

public BurgerMenu() {

menu.add(new MenuItem("ハンバーガー", 150));

menu.add(new MenuItem("チーズバーガー", 180));

menu.add(new MenuItem("ビッグマック", 410));

}

public ArrayList<MenuItem> getMenu() { return menu; }

}

**課題完成時の画面**

世界にはばたくECCフーズ！

ただいまM&Aで拡大中！！

どのメニューを表示しますか？（0：ECCコーヒー、1：ECCドーナツ、2：ECCバーガー、-1：終了）＞**2**

ハンバーガー：150円

チーズバーガー：180円

ビッグマック：410円

どのメニューを表示しますか？（0：ECCコーヒー、1：ECCドーナツ、2：ECCバーガー、-1：終了）＞

# **J2Kad25S「ECCフーズ（匿名クラス）」※実践編P.125～P.130、または検索**

1. 各Iteratorクラスを対応するMenuクラスの内部クラスにせよ。
2. ①で作成した内部クラスを匿名クラスにせよ。

* 各Iteratorクラスは内部クラス・匿名クラスにすると不要になるが、削除せずに残しておくこと（他の課題でエラーが出るので）。

ヒント：内部クラス・匿名クラスについては以下を参照、もしくは検索すること

・（もし教科書を持ってきていれば）実践編P.125～P.130

・J2Kad20D「内部クラス」、J2Kad20C「匿名クラス（無名クラス）」

**課題完成時の画面**

（J2Kad25Aと同じ）

# **J2Kad25X「ArrayListのイテレータ」※実践編P.108、または検索**

BurgerMenuのメニュー管理で使われているArrayList自体がイテレータに対応している。BurgerMenuの処理をこのイテレータを使った処理に変更せよ（MenuIteratorインターフェイスを通してArrayListのイテレータを操作する）。

// ArrayListのイテレータの使い方

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

while(it.hasNext()) {

// it.next()でMenuItemを取得

}

ヒント：オブジェクトアダプタ（委譲を利用したAdapterパターン）を適用する。

**課題完成時の画面**

（J2Kad25Aと同じ）