● 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」）

```
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"); }
}
```

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。

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

```
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
}
```

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。

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

```
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
}
```

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）。

● 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。

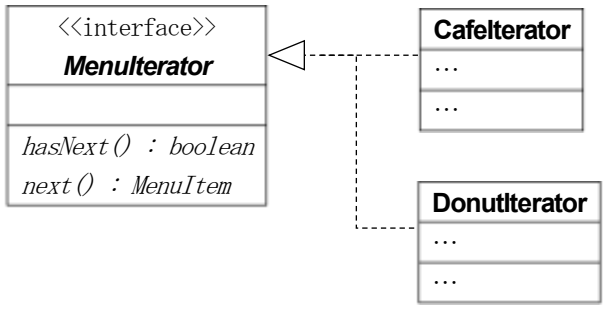
CafeIterator クラス（ファイル「CafeMenu.java」に作成）

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



課題完成時の画面

（J2Kad25D と同じ）

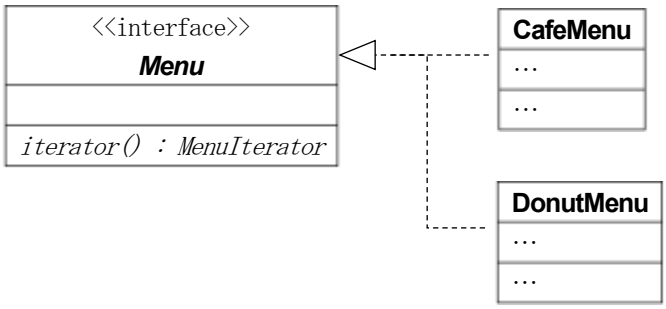
● J2Kad25B「ECC フーズ（Iterator パターン）」

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



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」）

```
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; }
}
```

ArrayList で管理している。

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）