● J2Challenge12A1「ECC ドーナツ！」

世界にはばたく ECC フーズがあのドーナツチェーンを買収した！その名も「ECC ドーナツ」。ECC ドーナツのメニューを表示する処理を作成せよ。なお、ECC ドーナツでは商品名と値段をそれぞれ String 型配列と int 型配列で管理している。

リスト１：ECC ドーナツのメニュー管理（DonutMenu クラス）

```
public class DonutMenu {
    private String[] names;
    private int[] price;

    public DonutMenu() {
        names = new String[100];          price = new int[100];
        names[0] = "ハニーディップ";      price[0] = 120;
        names[1] = "ハニーチュロ";        price[1] = 130;
        names[2] = "チョコリング";        price[2] = 140;
                                          price[3] = -1;       // 終了コード
    }
    public String toString() { return "ECC ドーナツ"; }
    public String[] getNames() { return names; }
    public int[] getPrice() { return price; }
}
```

なぜ商品名と値段を別々の配列で管理しているのかなどと言ってはいけない。
買収してフタを開けたらそうなっていたのだから仕方がない。

課題完成時の画面

```
世界にはばたく ECC フーズがドーナツチェーンを買収した！！
ECC ドーナツのメニューを表示します！
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
```

● J2Challenge12A2「ECC コーヒー！」

世界にはばたく ECC フーズがあのコーヒーチェーンを買収した！その名も「ECC コーヒー」。ECC コーヒーのメニューを表示する処理を作成せよ。なお、ECC コーヒーでは商品名と値段をセットにしたクラス（MenuItem クラス）を定義、それを ArrayList で管理している。

リスト 1：メニューを表すクラス（MenuItem クラス）

ECC コーヒーでは商品名と値段をセットにした MenuItem クラスを定義している

```
public class MenuItem {
    private String name;
    private int price;
    public MenuItem(String name, int price) {
        this.name = name;
        this.price = price;
    }
    public String toString() { return name; }
    public int getPrice() { return price; }
}
```

リスト 2：ECC コーヒーのメニュー管理（CafeMenu クラス）

MenuItem クラスを ArrayList に格納して管理している。

```
import java.util.ArrayList;

public class CafeMenu {
    private ArrayList<MenuItem> menu;
    public CafeMenu() {
        menu = new ArrayList<>();
        menu.add(new MenuItem("ドリップコーヒー", 390));
        menu.add(new MenuItem("アールグレイ", 430));
        menu.add(new MenuItem("オレンジジュース", 220));
    }
    public String toString() { return "ECC コーヒー"; }
    public ArrayList<MenuItem> getMenu() { return menu; }
}
```

課題完成時の画面

```
世界にはばたく ECC フーズがカフェチェーンを買収した！！
ECC コーヒーのメニューを表示します！
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
```

● J2Challenge12A3「ECC バーガー！」

世界にはばたく ECC フーズがあのハンバーガーチェーンを買収した！その名も「ECC バーガー」。ECC バーガーのメニューを表示する処理を作成せよ。なお、ECC バーガーでは商品名を key、値段を value とする HashMap を使って管理している。

リスト１：ECC バーガーのメニュー管理（BurgerMenu クラス）

HashMap（商品名が key、値段が value）で管理している。

```
import java.util.HashMap;

public class BurgerMenu {
    private HashMap<String, Integer> menu;
    public BurgerMenu() {
        menu = new HashMap<>() {
            {
                put("ハンバーガー", 150);
                put("チーズバーガー", 180);
                put("ビッグマック", 410);
            }
        };
    }
    public String toString() { return "ECC バーガー"; }
    public HashMap<String, Integer> getMenu() { return menu; }
}
```

課題完成時の画面

```
世界にはばたく ECC フーズがハンバーガーチェーンを買収した！！
ECC バーガーのメニューを表示します！
ビッグマック：410 円
チーズバーガー：180 円
ハンバーガー：150 円
```

● J2Challenge12S「メニュー表示を統合せよ！」

課題完成時の画面を参考に ECC ドーナツ、ECC コーヒー、ECC バーガーから選択した店のメニューを表示する処理を作成せよ。なお、これまでに学習した知識を動員して可能な限り簡略化されたエレガントなコードで記述すること。必要であればクラスやインターフェイスを追加してもよい。ただし各チェーン店のメニュー管理方法（2 つの配列で管理、ArrayList で管理、HashMap で管理）は変更できないものとする。

課題完成時の画面

0 で ECC ドーナツ、1 で ECC コーヒー、2 で ECC バーガーのメニューを表示する。-1（マイナスの値）が入力されたら終了、これら以外の値のときは ECC ドーナツを表示するものとする。

```
世界にはばたく ECC フーズ！
ただいま M&A で拡大中！！

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

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

どの店のメニューを表示しますか？（0：ECC ドーナツ、1：ECC コーヒー、2：ECC バーガー、-1：終了）＞2
ECC バーガーのメニューを表示します！
ビッグマック：410 円
チーズバーガー：180 円
ハンバーガー：150 円

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

● J2Kad12X「ペイントアルゴリズム！」

J2Kad08A「図形を描画しよう！」完成版相当に「5：ペイント」が追加されたプログラムが準備されている。Painter クラスのコンストラクタで指定された座標を起点に draw メソッドで塗りつぶしを行う。ただし draw メソッドから呼び出される Canvas クラスの paint メソッドが未実装なので何も起こらない。paint メソッドを実装して塗りつぶしできるようにせよ。

ヒント：座標を覚える LinkedList が必要になるのであらかじめ Point クラス（**リスト 3**）を作成しておくと便利

入力	描画する図形	使用するクラス
0	図形を消去する	Eraser
1	二等辺三角形を描画する。左下座標：(1, 22)、底辺の長さ：18、高さ：9	Triangle
2	長方形を描画する。左上座標：(12, 5)、横幅：20、高さ：14	Rectangle
3	円を描画する。中心座標：(29, 9)、半径：8	Circle
4	正方形を描画する。左上座標：(1, 1)、辺の長さ：16	Square
5	指定された座標(19, 11)を起点に塗りつぶしを行う。	Painter

リスト 1：Painter クラス

```
public class Painter implements IShape {
        ⋮
    public void draw(Canvas c) { c.paint(x, y); }
}
```

Canvas クラスの paint メソッドを呼び出しているが paint メソッドが未作成

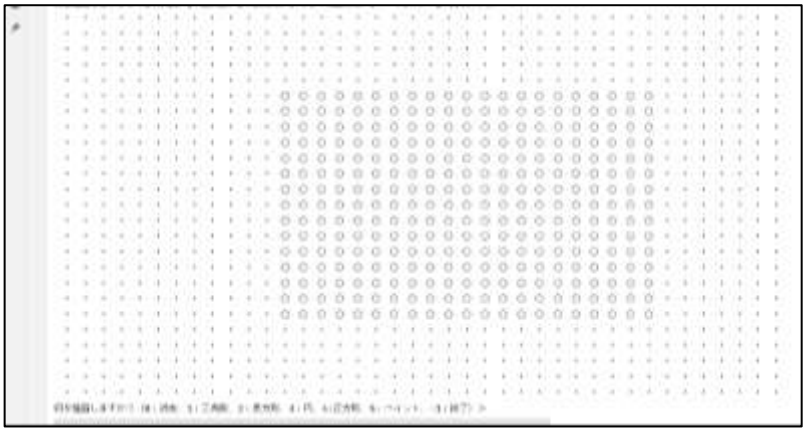
リスト 2：Canvas クラス

```
public class Canvas {
    private boolean[][] pixel;                    // true：点あり
        ⋮
    private boolean inBound(int x, int y) {…}     // true：描画領域内
    public void set(int x, int y) {…}             // 指定座標に点を打つ
    public void paint(int x, int y) {
        作成すること
    }
}
```

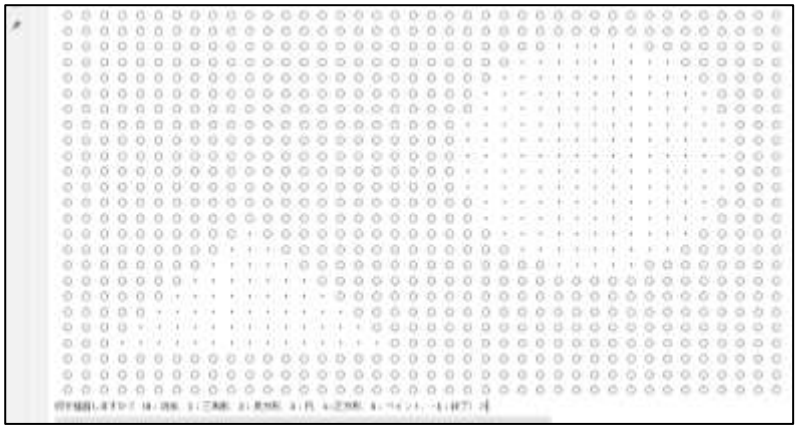
リスト 3：Point クラス（あらかじめ作成しておくと便利）

```
public class Point {
    public int x;
    public int y;
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
}
```

課題完成時の画面（画面のほぼ中央がペイントの起点）



2：長方形を実行してから 5：ペイントを実行
長方形の内部が塗りつぶされる。



1：二等辺三角形、3：円を実行してから
5：ペイントを実行
三角形と円の外側すべてが塗りつぶされる。

● ペイントアルゴリズム　←検索

ペイントアルゴリズムはたぶんいろいろとあると思いますが（検索すること）、以下、一例です。

① 座標を格納する LinkedList を 2 つ準備する（newList と oldList とする）。
② 起点が塗りつぶし可能（描画領域内かつ点が打たれていない）であれば、その座標に点を打って、座標を newList に追加する。
③ newList に座標がなければ終了。
④ oldList に newList をコピーして、newList は新規に LinkedList を new する。
⑤ oldList に座標がある間は以下を繰り返す。なければ③へ。
⑥ oldList から座標を取り出し、その座標の上下左右の点が塗りつぶし可能であれば、その座標に点を打って、座標を newList へ追加する。
⑦ ⑤へ戻る。

簡単に言えば、起点が塗りつぶし可能であれば、起点を塗りつぶす。さらにその上下左右の点が塗りつぶし可能であれば、そこも塗りつぶす。さらに塗りつぶした点の上下左右も塗りつぶし可能であれば、そこも塗りつぶす。これを塗りつぶせる点がなくなるまで繰り返す、ということです。