● J2Kad23D「ガチャガチャマシーンの不具合」※J2Kad23D は修正しなくても OK

あの世界的大ヒット作「ポケット Duck！」を制作した ECC ゲームスが今度はガチャガチャマシーンに進出することになった！コインを入れても返却ボタンを押せば戻ってくるという画期的な仕様だ。ところが業者にプログラムを発注（丸投げ）したところ、重大な欠陥が見つかった！なんとカプセルの残り個数が 0 になっても、そのまま動いてしまう！！（コインを入れてハンドルを回せるがカプセルが出てこない・・・）。業者のプログラムを修正して「売り切れ」の仕様を追加せよ。

リスト１：業者のプログラム（pac23d.GachaMachine クラス）

```
public class GachaMachine {
        ⋮
    public void showState() {
        System.out.println("カプセルの残り：" + count);
        if (state == NO_COIN) {
            System.out.println("コイン：なし");
        } else if(state == HAS_COIN) {
            System.out.println("コイン：あり");
        }
    }
    public void insertCoin() {
        if (state == NO_COIN) {
            System.out.println("コインを入れました！");
            setState(HAS_COIN);
        } else if (state == HAS_COIN) {
            System.out.println("これ以上コインが入らない！");
        }
    }
    public void turnHandle() {
        if (state == NO_COIN) {
            System.out.println("ハンドルが回りません！");
        } else if (state == HAS_COIN) {
            System.out.println("カプセルが出ました！");
            decCount();
            setState(NO_COIN);
        }
    }
    public void ejectCoin() {
        if (state == NO_COIN) {
            System.out.println("何も起こりません！");
        } else if (state == HAS_COIN) {
            System.out.println("コインが返却されました！");
            setState(NO_COIN);
        }
    }
}
```

「売り切れ」を追加するとどうなるのか、考えてみよう。すべてのメソッドに売り切れ時の対応を追加する必要がある。

Before：カプセルの残りが 0 でもハンドルを回して「カプセルが出ました！」と表示される

```
ガチャガチャをします！
カプセルの残り：3
コイン：なし
どうしますか？（0：コインを入れる、1：ハンドルを回す、2：返却ボタンを押す、-1：終わる）＞0
コインを入れました！
カプセルの残り：3
コイン：あり
どうしますか？（0：コインを入れる、1：ハンドルを回す、2：返却ボタンを押す、-1：終わる）＞1
カプセルが出ました！
カプセルの残り：2
      ⋮
どうしますか？（0：コインを入れる、1：ハンドルを回す、2：返却ボタンを押す、-1：終わる）＞1
カプセルが出ました！
カプセルの残り：0
コイン：なし
どうしますか？（0：コインを入れる、1：ハンドルを回す、2：返却ボタンを押す、-1：終わる）＞0
コインを入れました！
カプセルの残り：0
コイン：あり
どうしますか？（0：コインを入れる、1：ハンドルを回す、2：返却ボタンを押す、-1：終わる）＞1
カプセルが出ました！
カプセルの残り：-1
```

After：カプセルの残りが 0 になると「売り切れ！」と表示され、どの操作も受け付けなくなる

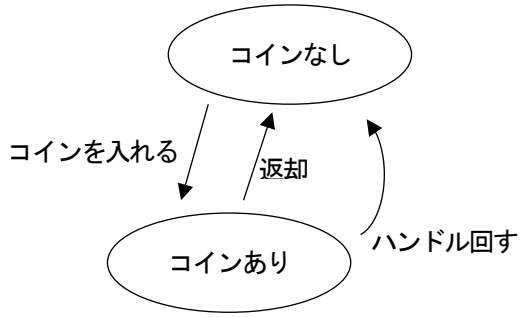
```
      ⋮
カプセルの残り：1
コイン：あり
どうしますか？（0：コインを入れる、1：ハンドルを回す、2：返却ボタンを押す、-1：終わる）＞1
カプセルが出ました！
カプセルの残り：0
売り切れ！
どうしますか？（0：コインを入れる、1：ハンドルを回す、2：返却ボタンを押す、-1：終わる）＞0
コイン投入口が閉まっています！
カプセルの残り：0
売り切れ！
どうしますか？（0：コインを入れる、1：ハンドルを回す、2：返却ボタンを押す、-1：終わる）＞1
ハンドルが回りません！
カプセルの残り：0
売り切れ！
どうしますか？（0：コインを入れる、1：ハンドルを回す、2：返却ボタンを押す、-1：終わる）＞2
何も起こりません！
カプセルの残り：0
売り切れ！
どうしますか？（0：コインを入れる、1：ハンドルを回す、2：返却ボタンを押す、-1：終わる）＞-1
```

● J2Kad23C「状態クラス」

※ここからは src フォルダ直下のファイル「GachaMachine.java」を修正すること

ガチャガチャマシーンの状態を表す GachaState インターフェイス、NoCoin クラス、HasCoin クラスを作成し、動作確認せよ。

ガチャガチャマシーンの状態遷移図



J2Kad23C で作成するのは〇の部分
矢印の処理は J2Kad23B で作成する

GachaState インターフェイス（ファイル「GachaMachine.java」に作成）

＜＜interface＞＞ ***GachaState***
showState() : void *insertCoin(gm : GachaMachine) : void* *turnHandle(gm : GachaMachine) : void* *ejectCoin(gm : GachaMachine) : void*

状態クラスの仕様（GachaState インターフェイスを実装、ファイル「GachaMachine.java」に作成する）

クラス	状態の表示 showState	コインを入れる insertCoin	ハンドルを回す turnHandle	返却ボタンを押す ejectCoin
NoCoin	コイン：なし	コインを入れました！	ハンドルが回りません！	何も起こりません！
HasCoin	コイン：あり	これ以上コインが入りません！	カプセルが出ました！	コインを返却しました！

リスト１：状態クラスの動作確認（J2Kad23C クラス）

```
public class J2Kad23C {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("ガチャガチャをします！");
        GachaMachine gm = new GachaMachine();
        while(true) {
            System.out.print("どの状態をチェックしますか？ (0：NoCoin、1：HasCoin、-1：終わる) ＞");
            int n = Integer.parseInt(in.next());
            if (n < 0) break;
            選択した状態クラス（NoCoin またはHasCoin）を生成し、
            showState、insertCoin、turnHandle、ejectCoin を順番に実行する。
            System.out.println();
        }
    }
}
```

課題完成時の画面

```
どの状態をチェックしますか？ (0：NoCoin、1：HasCoin、-1：終わる) ＞0
コイン：なし
コインを入れました！
ハンドルが回りません！
何も起こりません！

どの状態をチェックしますか？ (0：NoCoin、1：HasCoin、-1：終わる) ＞1
コイン：あり
これ以上コインが入りません！
カプセルが出ました！
コインを返却しました！

どの状態をチェックしますか？ (0：NoCoin、1：HasCoin、-1：終わる) ＞-1
```

● J2Kad23B「状態遷移（State パターン）」※J2Kad23B は修正しなくても OK

J2Kad23B にガチャガチャマシーンを操作する処理が準備されている。

① NoCoin クラスと HasCoin クラスに状態遷移処理を追加せよ。
② GachaMachine に現在の状態を表すフィールド state を追加し、showState、insertCoin、turnHandle、ejectCoin の各メソッドから state の対応するメソッドへ委譲する処理を追加せよ。

状態遷移表

クラス	コインを入れる insertCoin	ハンドルを回す turnHandle	返却ボタンを押す ejectCoin
NoCoin	表示：コインを入れました！ **状態：HasCoin へ遷移**	表示：ハンドルが回りません！ **状態：遷移なし**	表示：何も起こりません！ **状態：遷移なし**
HasCoin	表示：これ以上コインが入りません！ **状態：遷移なし**	表示：カプセルが出ました！ **処理：カプセルを減らす** **状態：NoCoin へ遷移**	表示：コインを返却しました！ **状態：NoCoin へ遷移**

リスト１：状態クラスへの委譲（GachaMachine クラス）

```
// ガチャガチャマシーン
public class GachaMachine {
        ⋮
    private GachaState state = new NoCoin();                    // 状態、最初は NoCOin
    public void setState(GachaState state) { this.state = state; }    // 状態を切り換える
    public void showState() {
        System.out.println("カプセルの残り：" + count);
        現在の状態の showSate へ委譲
    }
    public void insertCoin() { 現在の状態の insertCoin へ委譲 }
    public void turnHandle() { 現在の状態の turnHandle へ委譲 }
    public void ejectCoin() { 現在の状態の ejectCoin へ委譲 }
}
```

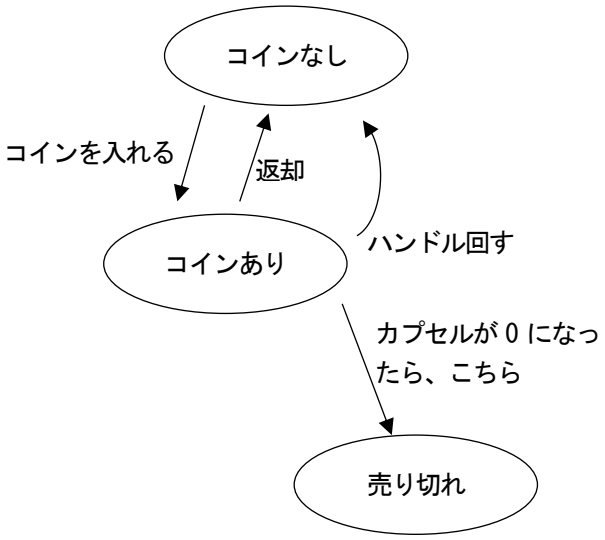
追加

課題完成時の画面

（J2Kad23D の Before と同じ）

● J2Kad23A「ガチャガチャマシーン完成！」※J2Kad23A は修正しなくても OK

ガチャガチャマシーンに「売り切れ」（SoldOut クラス）を追加せよ。なお、売り切れ時の状態表示（showState メソッド）は「売り切れ！」と表示すること。



状態遷移表（SoldOut を追加）

クラス	コインを入れる insertCoin	ハンドルを回す turnHandle	返却ボタンを押す ejectCoin
NoCoin	表示：コインを入れました！ 状態：HasCoin へ遷移	表示：ハンドルが回りません！ 状態：遷移なし	表示：何も起こりません！ 状態：遷移なし
HasCoin	表示：これ以上コインが入りません！ 状態：遷移なし	表示：カプセルが出ました！ 処理：カプセルを減らす 状態： **カプセルが残っていれば NoCoin** **残っていなければ SoldOut へ**	表示：コインを返却しました！ 状態：NoCoin へ遷移
SoldOut	**表示：コイン投入口が閉まっています！** **状態：遷移なし**	**表示：ハンドルが回りません！** **状態：遷移なし**	**表示：何も起こりません！** **状態：遷移なし**

課題完成時の画面

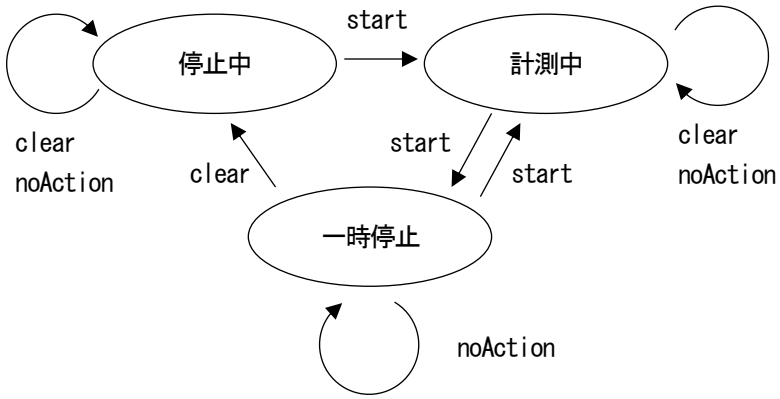
（J2Kad23D の After と同じ）

● J2Kad23S「ストップウォッチ！（State 版）」※J2Kad23S は修正しなくても OK

状態遷移図をもとにストップウォッチの処理を作成せよ。操作は「0：START」と「1：CLEAR」の 2 つとし、マイナスの値を入力すると終了、それ以外の値を入力すると何も操作しない状態（noAction）とする。

※　ここで作成するのは状態遷移のみです。実際の時間計測は行いません。

ストップウォッチの状態遷移図



ヒント：
状態遷移図から状態遷移表を作ってみること。あとは状態遷移表にもとづいてクラスを作成し、状態遷移表のすべての項目が仕様通りに動作するかどうかをチェックする。

課題完成時の画面

```
状態：停止中
どうしますか？（0：START、1：CLEAR、-1：終了）＞2
止まっています・・・
状態：停止中
どうしますか？（0：START、1：CLEAR、-1：終了）＞1
何も起こりません！
状態：停止中
どうしますか？（0：START、1：CLEAR、-1：終了）＞0
計測を始めます！
状態：計測中
どうしますか？（0：START、1：CLEAR、-1：終了）＞2
計測中です・・・
状態：計測中
どうしますか？（0：START、1：CLEAR、-1：終了）＞1
何も起こりません！
状態：計測中
```

（続き）

```
どうしますか？（0：START、1：CLEAR、-1：終了）＞0
一時停止します！
状態：一時停止中
どうしますか？（0：START、1：CLEAR、-1：終了）＞2
一時停止中です・・・
状態：一時停止中
どうしますか？（0：START、1：CLEAR、-1：終了）＞0
計測を再開します！
状態：計測中
どうしますか？（0：START、1：CLEAR、-1：終了）＞0
一時停止します！
状態：一時停止中
どうしますか？（0：START、1：CLEAR、-1：終了）＞1
タイムをリセットして停止します！
状態：停止中
どうしますか？（0：START、1：CLEAR、-1：終了）＞-1
```

● J2Kad23X「世界にはばたく ECC フーズ！」※次回解答編

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 ドーナツと ECC コーヒー、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。全チェーン店のメニューを表示するシステムを作成せよ。なお DonutMenu と CafeMenu にメソッドの追加や修正を行っても良いが、管理方法（データ構造）の変更は不可とする。

メニュー管理の方法

店名	管理方法（変更不可）
ECC ドーナツ（DonutMenu クラス）	ドーナツの名前と値段を String 型の配列と int 型の配列で管理
ECC コーヒー（CafeMenu クラス）	MenuItem クラスの ArrayList で管理

※　可能な限りエレガントなコードを記述すること。なお、本課題は今回のテーマ（State）とは関係ないので注意すること（ヒント参照）。

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 ドーナツ、1：ECC コーヒー、-1：終了）>0
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 ドーナツ、1：ECC コーヒー、-1：終了）>1
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 ドーナツ、1：ECC コーヒー、-1：終了）>-1
```

ヒント：
「Iterator パターン」（←検索、次回予定）を適用するとエレガントになる（**リスト 2**）。ただし、わからないときはべたべたのコード（**リスト 1**）でも動作していれば本課題は OK とする。

リスト 1：べたべたバージョン

```
if (shop == 0) {
    DonutMenu クラスの生成
    ドーナツメニューの表示
} else if (shop == 1) {
    CafeMenu クラスの生成
    コーヒーメニューの表示
}
```

リスト 2：こういうふうにしたいバージョン

```
if (shop == 0) {
    DonutMenu 関連クラスの生成
} else if (shop == 1) {
    CafeMenu 関連クラスの生成
}
メニューの表示（共通処理）
```

でもこれだけで満足しないこと…