● ガチャガチャマシーンの不具合（課題ではありません。これは J2Kad24D～A までの説明です）

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。ところが完成版をリリースしようとしたところ、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・・・）。そこで「売り切れ」の仕様を追加しようとしたところ、業者のプログラム（もちろん業者に丸投げです）では修正が多岐にわたってしまう！State パターンを使ってリファクタリングせよ。

業者のプログラムの実行画面（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！」と表示する）

```
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
```

リスト１：業者のプログラム（pac24b.GachaMachine クラス）

J2Kad24B（pac24b）のファイル「GachaMachine.java」が業者のプログラム

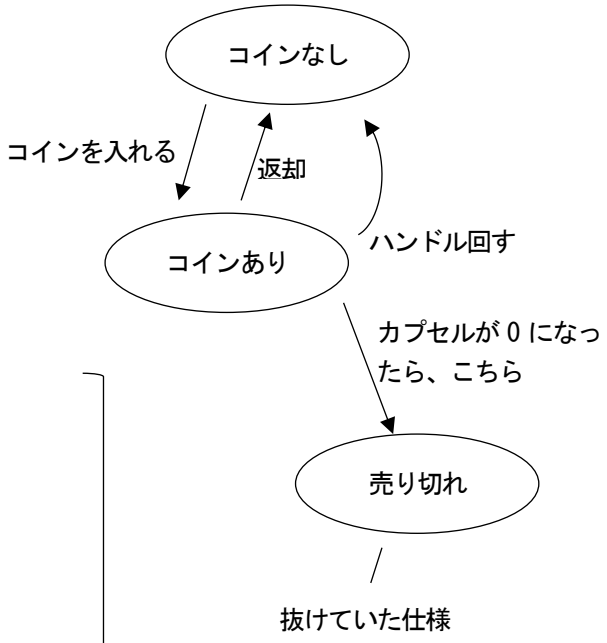
```
public class GachaMachine {
    private final int NO_COIN = 0;              // コインなし
    private final int HAS_COIN = 1;             // コインあり
    private int state = NO_COIN;                // 現在の状態
    private int count = 3;                      // カプセルの数

    public void setState(int newState) { state = newState; }    // 状態の切り替え
    public int decCount() { return --count; }                   // カプセルの数を減らす（戻り値は減った後の数）

    public void execute() {
        Scanner in = new Scanner(System.in);
```

リスト１：業者のプログラム（続き）

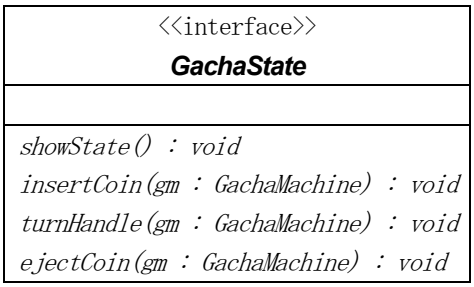
```
        while(true) {
            System.out.println("カプセルの残り：" + count);
            showState();
            System.out.print("どうしますか？ (0：コインを入れる、1：ハンドルを回す、2：返却ボタンを押す、-1：
終わる) ＞");
            int cmd = Integer.parseInt(in.next());
            if (cmd < 0) break;
            switch(cmd) {
                case 0: insertCoin();   break;
                case 1: turnHandle();   break;
                case 2: ejectCoin();    break;
            }
        }
    }
    private void showState() {
        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");
        }
    }
    private 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！");
        }
    }
    private 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();                         // カプセル減らす
            setState(NO_COIN);
        }
    }
    private 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。

● J2Kad24D「状態クラス」

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、GachaMachine.execute メソッドで動作確認せよ。



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（GachaMachine クラス）

```
public class GachaMachine {
    private GachaState state;
    public void execute() {
        Scanner in = new Scanner(System.in);
        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;
            System.out.println();
        }
    }
}
// GachaState インターフェイス
作成すること
// NoCoin クラス
作成すること
// HasCoin クラス
作成すること
```

追加する

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。

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
```

● J2Kad24C「状態遷移」※J2Kad24D の GachaState、NoCoin、HasCoin をコピーすること

J2Kad24D で作成した NoCoin クラスと HasCoin クラスに状態遷移処理を追加せよ。

※ J2Kad24D の GachaState インターフェイス、NoCoin クラス、HasCoin クラスをファイル「GachaMachine.java」にコピーして作成する。ただし**貼り付ける場所で右クリックして[特殊なコピー/貼り付け]→[プレーンテキストとして貼り付け]**で貼り付けること。

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！ **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：状態クラスの動作確認（GachaMachine クラス）

```
public class GachaMachine {
    private GachaState state;
    public void setState(GachaState state) { this.state = state; }
    public void execute() {
        Scanner in = new Scanner(System.in);
        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;
            System.out.println();
        }
    }
}
// GachaState インターフェイス
J2Kad24D からコピー
// NoCoin クラス
J2Kad24D からコピー
// HasCoin クラス
J2Kad24D からコピー
```

追加する

選択した状態クラス（NoCoin または HasCoin）を生成し、以下の順番で処理を実行する。
showState、insertCoin、showState、turnHandle、showState、ejectCoin

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！
コイン：あり
カプセルが出ました！
コイン：なし
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！
コイン：あり
カプセルが出ました！
コイン：なし
何も起こりません！
```

状態が変化するのを確認する

● J2Kad24B「リファクタリング！」※J2Kad24C の GachaState、NoCoin、HasCoin をコピー

パッケージ pac24b に業者が作った状態の GachaMachine が準備されている。J2Kad24C の GachaState、NoCoin、HasCoin をコピーして、State パターンを使ってリファクタリングせよ。なお、現時点ではまだ「売り切れ」に未対応で構わない（「売り切れ」への対応は次の課題で行う）。

課題完成時の画面（完成前と完成後で動作は同じ、内部の設計が異なる）

```
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）＞1
ハンドルが回りません！
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）＞2
何も起こりません！
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）＞0
これ以上コインが入りません！
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
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：2
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）＞2
コインを返却しました！
カプセルの残り：2
コイン：なし
      ⋮
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
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）＞-1
```

J2Kad24C の「状態遷移表」のすべての項目が正常に動作しているか確認すること。なお、「売り切れ」は未実装なので、カプセル数が 0 でもカプセルを出す処理を実行してしまう。

● J2Kad24A「ガチャガチャマシーン完成！」※J2Kad24B の「GachaMachine.java」をコピー

パッケージ pac24a にはダミーの「GachaMachine.java」が入っている。パッケージ pac24b の「GachaMachine.java」を上書きコピーし、「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

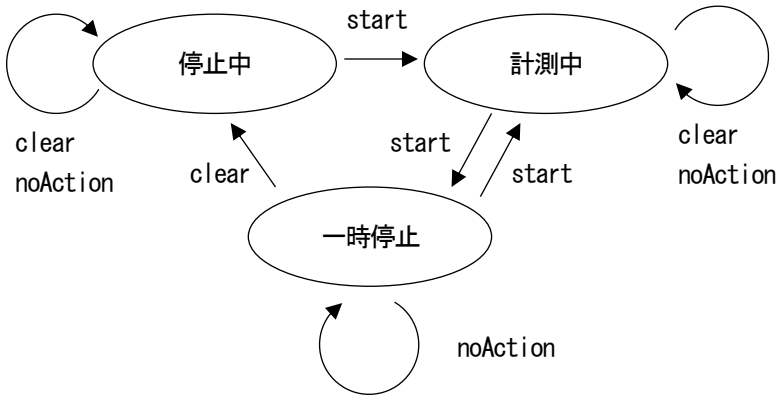
```
ガチャガチャをします！
      ⋮
カプセルの残り：1
コイン：あり
どうしますか？（0：コインを入れる、1：ハンドルを回す、2：返却ボタンを押す、-1：終わる）＞1
カプセルが出ました！
カプセルの残り：0
売り切れ！
どうしますか？（0：コインを入れる、1：ハンドルを回す、2：返却ボタンを押す、-1：終わる）＞0
コイン投入口が閉まっています！
カプセルの残り：0
売り切れ！
どうしますか？（0：コインを入れる、1：ハンドルを回す、2：返却ボタンを押す、-1：終わる）＞1
ハンドルが回りません！
カプセルの残り：0
売り切れ！
どうしますか？（0：コインを入れる、1：ハンドルを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

● J2Kad24S「ストップウォッチ！（State 版）」※JKad23X（前期課題）の State パターン版

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基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
```

● J2Kad24X「世界にはばたく ECC フーズ！」※J2Challenge12S のリマインド、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）	ドーナツの名前を String 型の配列、値段を 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（仕様のすべての動作に問題がないか確認すること）

```
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①：
「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）
```

ヒント②：
（もし教科書を持ってきていたら）実践編 P.108「イテレータ」にイテレータのしくみの記述あり。