# **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）**

「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。

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);

}

}

}

**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○の部分

矢印の処理はJ2Kad23Bで作成する

コインを入れる

返却

ハンドル回す

**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;

System.out.println();

}

}

}

**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**

# **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。

コインを入れる

返却

ハンドル回す

カプセルが0になったら、こちら

**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**

clear

noAction

start

start

clear

start

noAction

clear

noAction

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 （続き）**

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

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 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）**

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**

}

でもこれだけで満足しないこと…