【サンプルについて】

Flashで自販機（疑似ハードウェア）を造ってみました
Flash（ActionScript）は高級言語なので、誌上で説明した内容と若干異なっている所もありますが
極めて単純な構成で出来てしまうことは理解いただけると思います

ーサンプル構成ー

［ハードウェア］
　Flash版の疑似ハードウェアです
　（FlashPlayerがなくても動作できるようプロジェクタ形式になっています
　　Windows用とMac用があります）
・缶ジュース自販機.exeまたは.app
・たばこ自販機.exeまたは.app
・券売機.exeまたは.app
［フレームワーク］
　組込みではメモリー上に配置されるためツール内で隠蔽されているデータですが
　今回は読んだり変更したりできるようXML形式にしてあります
　（エンコーディングはUTF-8です。UTF-8の扱えるエディタで開いて下さい）
・STAFF_UTF.xml
　　コンストラクタテーブルです
　　（コンストラクタへの引数の内容はソースを参照して下さい）
・SCEN_UTF.xml
　　スレッドテーブルです
［コンポーネント］
　独立性がわかるようにクラス定義をひとつずつのMovieClip（.swf）にしてあります
　本体である疑似ハードウェアが起動時に読込んでインスタンスを生成して動作します

ー動作手順ー

・ハードウェアから自販機のどれかを選びます
・選択した自販機に対応するSTAFF_UTF.xmlとSCEN_UTF.xmlを選びます
・コンポーネントは面倒なので（使わないものもありますが）すべて選択します
・これらを同じフォルダーに入れて自販機をダブルクリックすれば起動します
・自販機の操作は右図の通りです

※外部の.swfを読込む方式はマシンとの相性があるようです
　下部のアイコンマーチが欠けてしまう場合は正しく動作していません（もしかして動かなかったらごめんなさいです）

－参考－

文末にこのサンプルのＳＣＯＲＥ表記シナリオと
各コンポーネントのソース（ActionScript）を掲げておきました

－連絡先－

質問、ご意見あるいは苦情等ありましたら以下へお願いします

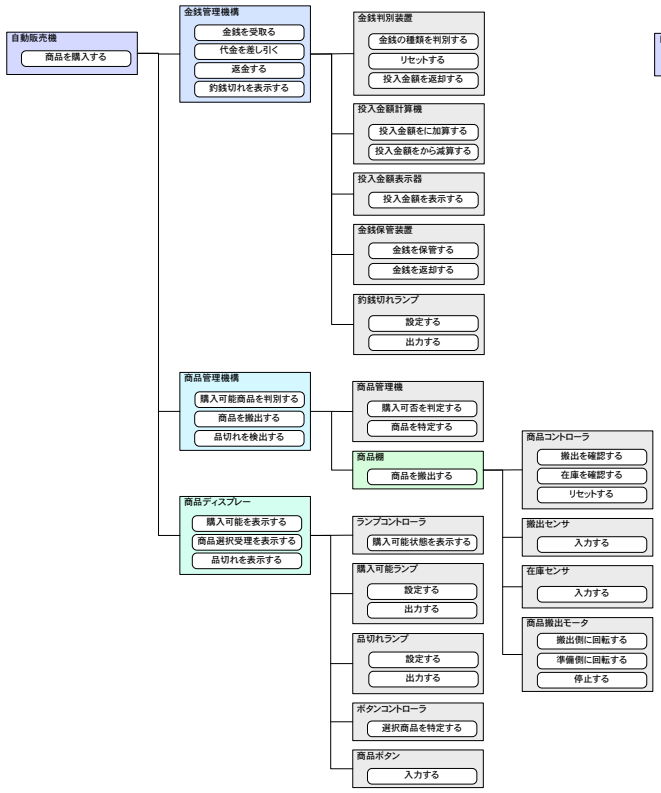
k_natsu@mac.com

ＳＣＯＲＥ

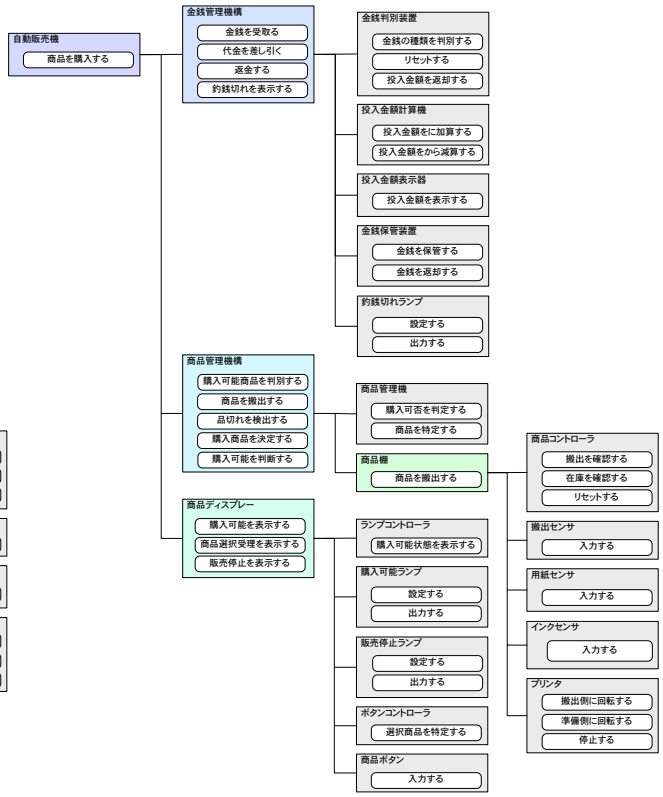
［役割構成］

缶ジュース販売機をベースに、券売機は金銭投入前に商品を選択するように
たばこ販売機は都度精算を一括精算に変えただけのバリエーションなので
役割構成の違いはほとんどありません

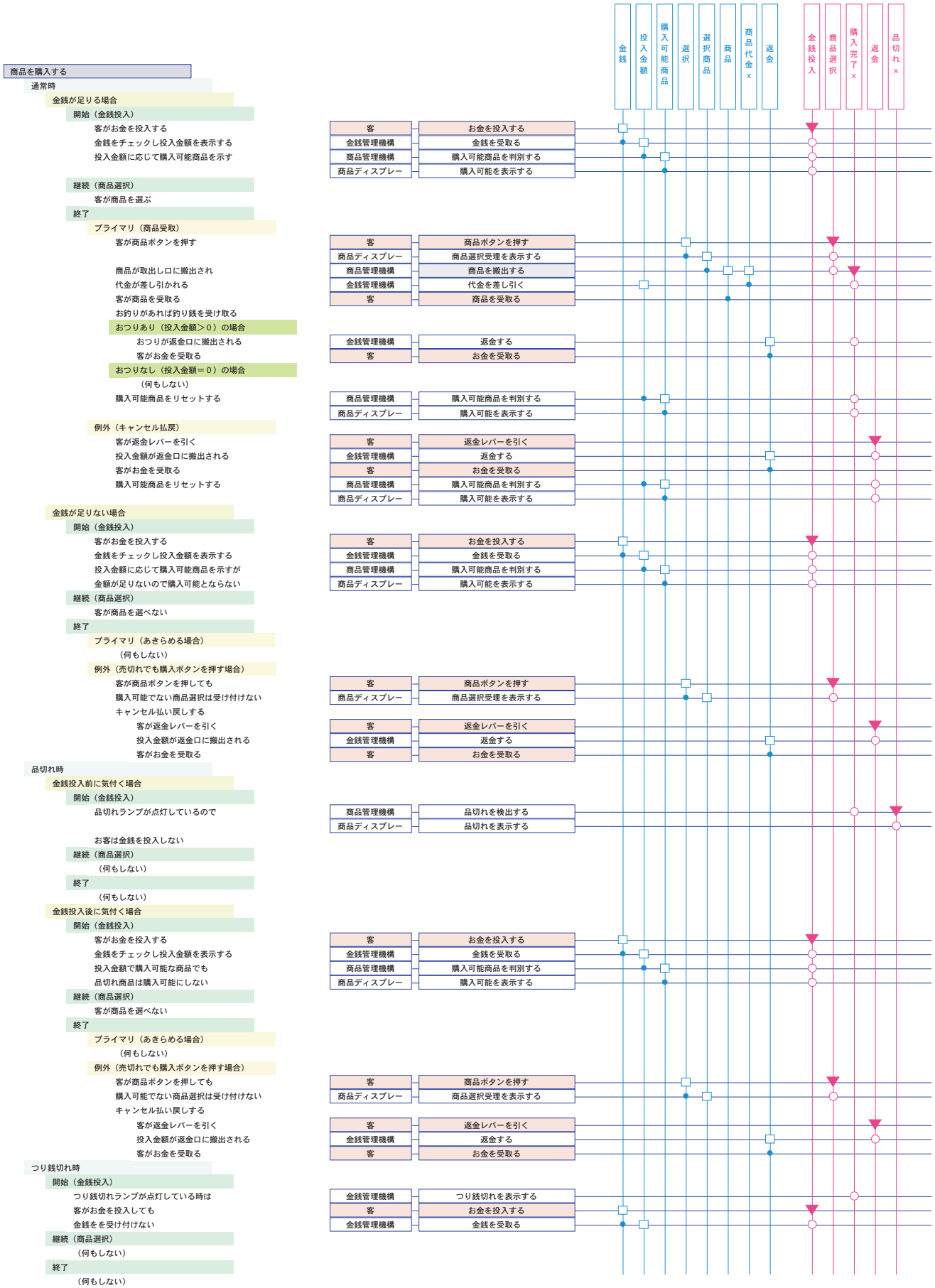
【缶ジュース販売機】　【たばこ販売機】



【券売機】



［缶ジュース販売機］–ルート–



商品を購入する

通常時

金銭が足りる場合

開始（金銭投入）

客がお金を投入する
金銭をチェックし投入金額を表示する
投入金額に応じて購入可能商品を示す

継続（商品選択）

客が商品を選ぶ

終了

プライマリ（商品受取）

客が商品ボタンを押す

商品が取出し口に搬出され
代金が差し引かれる
客が商品を受取る
お釣りがあれば釣り銭を受け取る

おつりあり（投入金額＞0）の場合

おつりが返金口に搬出される
客がお金を受取る

おつりなし（投入金額＝0）の場合

（何もしない）

購入可能商品をリセットする

例外（キャンセル払戻）

客が返金レバーを引く
投入金額が返金口に搬出される
客がお金を受取る
購入可能商品をリセットする

金銭が足りない場合

開始（金銭投入）

客がお金を投入する
金銭をチェックし投入金額を表示する
投入金額に応じて購入可能商品を示すが
金額が足りないので購入可能とならない

継続（商品選択）

客が商品を選べない

終了

プライマリ（あきらめる場合）

（何もしない）

例外（売切れでも購入ボタンを押す場合）

客が商品ボタンを押しても
購入可能でない商品選択は受け付けない
キャンセル払い戻しする

客が返金レバーを引く
投入金額が返金口に搬出される
客がお金を受取る

品切れ時

金銭投入前に気付く場合

開始（金銭投入）

品切れランプが点灯しているので

お客は金銭を投入しない

継続（商品選択）

（何もしない）

終了

（何もしない）

金銭投入後に気付く場合

開始（金銭投入）

客がお金を投入する
金銭をチェックし投入金額を表示する
投入金額で購入可能な商品でも
品切れ商品は購入可能にしない

継続（商品選択）

客が商品を選べない

終了

プライマリ（あきらめる場合）

（何もしない）

例外（売切れでも購入ボタンを押す場合）

客が商品ボタンを押しても
購入可能でない商品選択は受け付けない
キャンセル払い戻しする

客が返金レバーを引く
投入金額が返金口に搬出される
客がお金を受取る

つり銭切れ時

開始（金銭投入）

つり銭切れランプが点灯している時は
客がお金を投入しても
金銭をを受け付けない

継続（商品選択）

（何もしない）

終了

（何もしない）

–金銭管理機構–

金銭を受取る

金銭をチェックし投入金額を表示する

つり銭切れランプが点灯している時は金銭を受け付けない

金銭受理の準備をする

つり銭切れでなければ金銭を受け付ける

投入された金銭の種類を判別する

有効な金銭の場合

投入金額に加算する

投入金額を表示する

投入された金銭を保管する

無効な金銭の場合

投入された金銭を返却する

代金を差し引く

投入金額から商品代金を差し引く

返金する

投入金額を返金口に搬出する

金額に応じた硬貨を返金口に搬出する

投入金額＝０にする

投入金額＝０を表示する

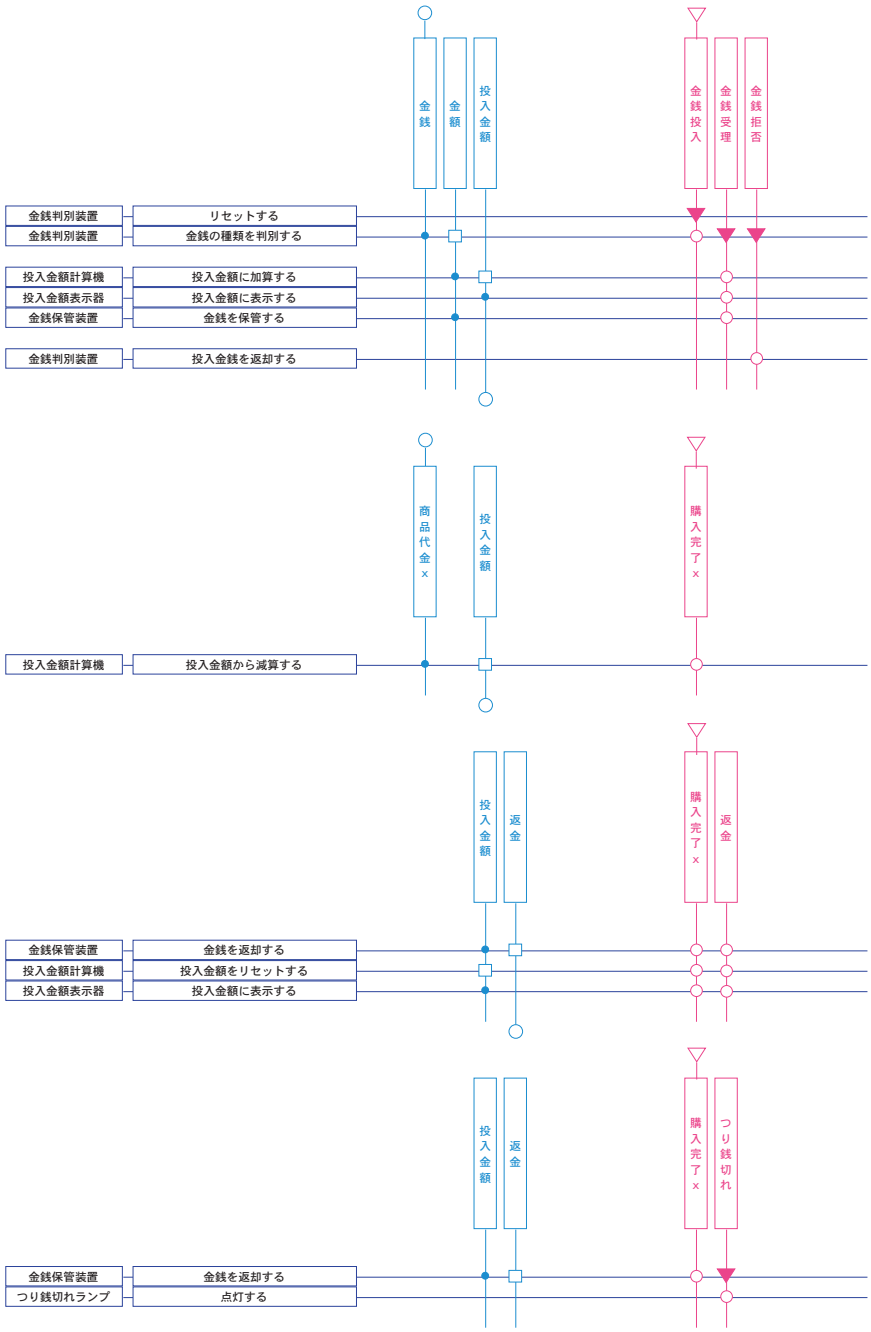
つり銭切れを表示する

つり銭切れ時はつり銭切れランプを点灯する

（※ランプ点灯時は金銭をを受け付けないので復帰はしない）

返金後、保有硬貨数が規定数以下になったら

つり銭切れランプを点灯する

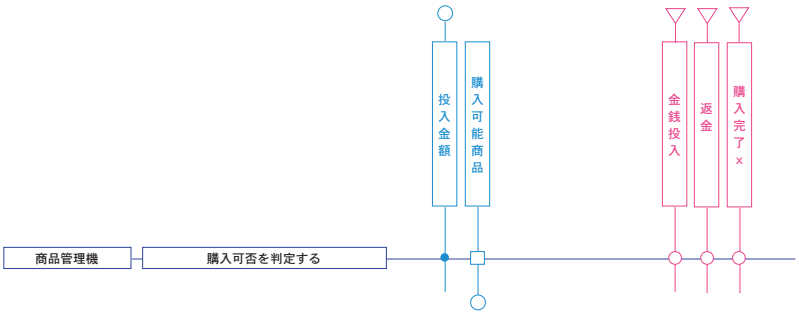


–商品管理機構–

購入可能商品を判別する

投入金額に応じて購入可能商品を示す

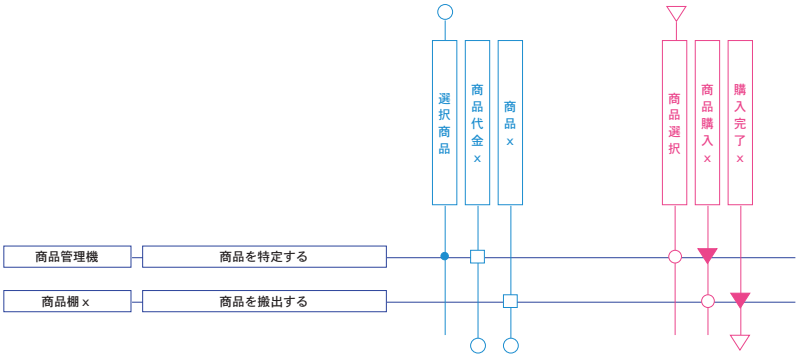
商品毎に購入可能か判定する



商品を搬出する

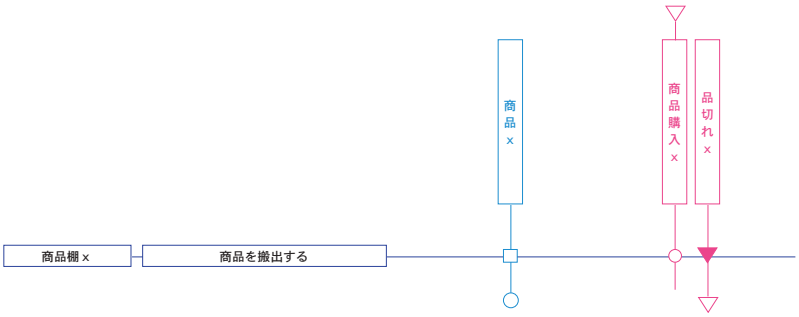
選択された商品を取出し口に搬出する

選択された商品の代金を取得し
格納している商品棚を特定して
商品を搬出する



品切れを検出する

最後の商品を搬出したら品切れを通知する



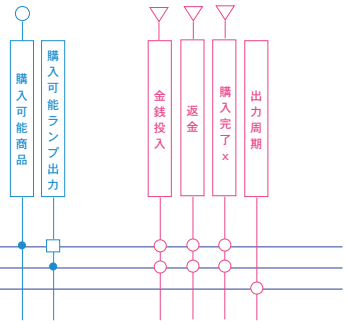
–商品ディスプレー–

購入可能を表示する

購入可能商品を示す

購入可能な商品のランプを点灯
‥購入不可能な商品のランプを消灯する

ランプコントローラ	購入可能状態を表示する
購入可能ランプ x	設定する
購入可能ランプ x	出力する

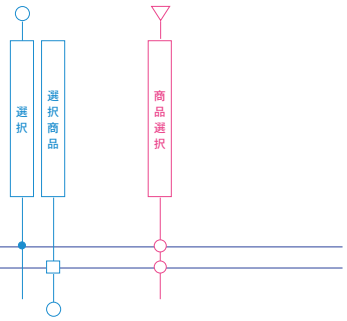


商品選択受理を表示する

商品ボタンが押されたことを判定する
購入可能でない商品選択は受け付けない

商品ボタンの押下を検出し
選択商品を特定する

商品ボタン x	入力する
ボタンコントローラ	選択商品を特定する

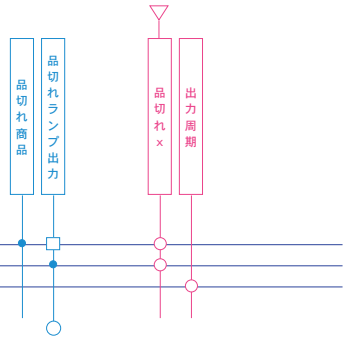


品切れを表示する

品切れランプを点灯する

品切れ商品の品切れランプを点灯する

ランプコントローラ	品切れを表示する
品切れランプ x	設定する
品切れランプ x	出力する



–商品棚–

商品を搬出する

プライマリ（商品が搬出できる場合）

搬出モータを回転して商品を搬出する

搬出が確認できたら、購入完了を通知する

搬出モータを回転して次の搬出を準備する
搬出準備が出来たらモータを止める

最後の１つを搬出したら、売切れを通知する

搬出モータ x	搬出側に回転する
商品コントローラ x	リセットする
搬出センサ x	入力する
商品コントローラ x	搬出を確認する
搬出モータ x	準備側に回転する
商品コントローラ x	リセットする
在庫センサ x	入力する
商品コントローラ x	在庫を確認する
搬出モータ x	停止する

例外（搬出故障の場合）

商品を搬出しようとしても

搬出が確認できないので購入完了しない
モータ保護のためタイムアウトで停止する

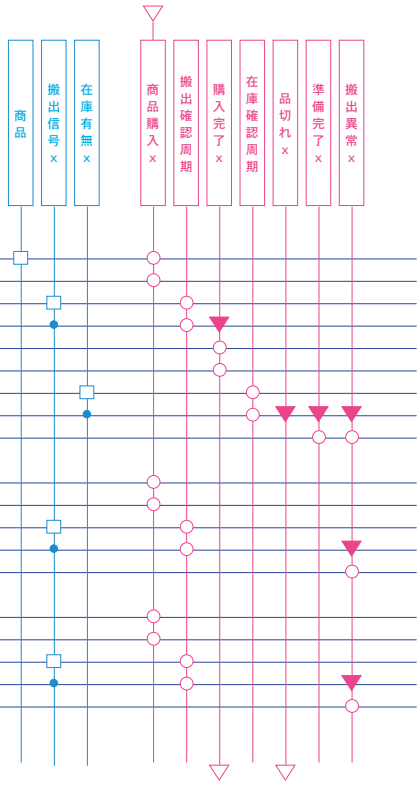
搬出モータ x	搬出側に回転する
商品コントローラ x	リセットする
搬出センサ x	入力する
商品コントローラ x	搬出を確認する
搬出モータ x	停止する

例外（売切れの場合）

（商品選択を受付けてないはずなので、これは起きない）
商品を搬出しようとしても

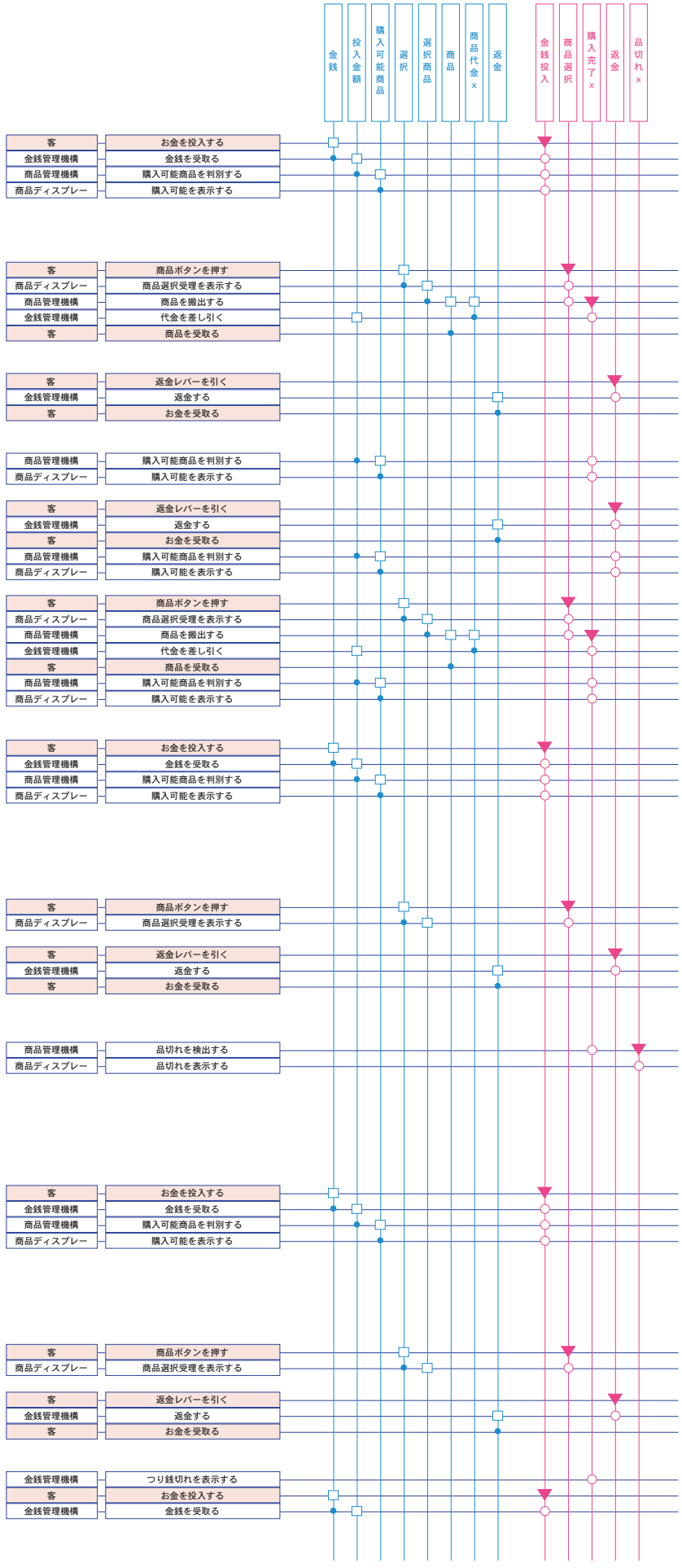
搬出が確認できないので購入完了しない
モータ保護のためタイムアウトで停止する

搬出モータ x	搬出側に回転する
商品コントローラ x	リセットする
搬出センサ x	入力する
商品コントローラ x	搬出を確認する
搬出モータ x	停止する



［たばこ販売機］–ルート–

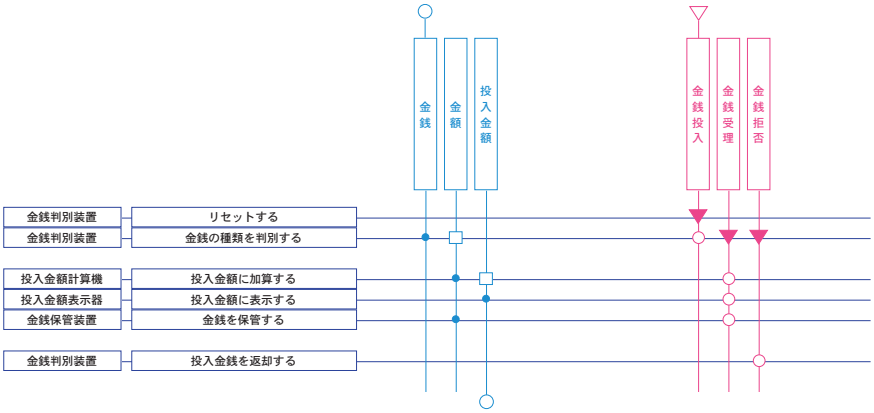
- 商品を購入する
  - 通常時
    - 金銭が足りる場合
      - 開始（金銭投入）
        - 客がお金を投入する
        - 金銭をチェックし投入金額を表示する
        - 投入金額に応じて購入可能商品を示す
      - 継続（商品選択）
        - 客が商品を選ぶ
      - 終了
        - プライマリ（商品受取して終了）
          - 客が商品ボタンを押す
          - 商品が取出し口に搬出され
          - 代金が差し引かれる
          - 客が商品を受取る
          - お釣りがあれば釣り銭を受け取る
          - おつりあり（投入金額＞０）の場合
            - 客が返金レバーを引く
            - 投入金額が返金口に搬出される
            - 客がお金を受取る
          - おつりなし（投入金額＝０）の場合
            - （何もしない）
          - 購入可能商品をリセットする
        - 例外（キャンセル払戻）
          - 客が返金レバーを引く
          - 投入金額が返金口に搬出される
          - 客がお金を受取る
          - 購入可能商品をリセットする
        - 例外（連続購入のため終了しない）
          - 客が商品ボタンを押す
          - 商品が取出し口に搬出され
          - 代金が差し引かれる
          - 客が商品を受取る
          - 購入可能商品を再表示する
    - 金銭が足りない場合
      - 開始（金銭投入）
        - 客がお金を投入する
        - 金銭をチェックし投入金額を表示する
        - 投入金額に応じて購入可能商品を示すが
        - 金額が足りないので購入可能とならない
      - 継続（商品選択）
        - 客が商品を選べない
      - 終了
        - プライマリ（あきらめる場合）
          - （何もしない）
        - 例外（金銭不足でも購入ボタンを押す場合）
          - 客が商品ボタンを押しても
          - 購入可能でない商品選択は受け付けない
        - キャンセル払い戻しする
          - 客が返金レバーを引く
          - 投入金額が返金口に搬出される
          - 客がお金を受取る
  - 品切れ時
    - 金銭投入前に気付く場合
      - 開始（金銭投入）
        - 品切れランプが点灯しているので
        - お客は金銭を投入しない
      - 継続（商品選択）
        - （何もしない）
      - 終了
        - （何もしない）
    - 金銭投入後に気付く場合
      - 開始（金銭投入）
        - 客がお金を投入する
        - 金銭をチェックし投入金額を表示する
        - 投入金額で購入可能な商品でも
        - 品切れ商品は購入可能にしない
      - 継続（商品選択）
        - 客が商品を選べない
      - 終了
        - プライマリ（あきらめる場合）
          - （何もしない）
        - 例外（売切れでも購入ボタンを押す場合）
          - 客が商品ボタンを押しても
          - 購入可能でない商品選択は受け付けない
        - キャンセル払い戻しする
          - 客が返金レバーを引く
          - 投入金額が返金口に搬出される
          - 客がお金を受取る
  - つり銭切れ時(何もしない)
    - 開始（金銭投入）
      - つり銭切れランプが点灯している時は
      - 客がお金を投入しても
      - 金銭をを受け付けない
    - 継続（商品選択）
      - （何もしない）
    - 終了



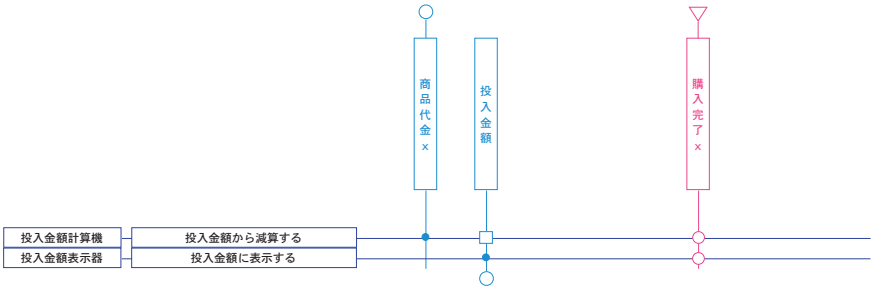
–金銭管理機構–

金銭を受取る
　金銭をチェックし投入金額を表示する
　つり銭切れランプが点灯している時は金銭を受け付けない

　金銭受理の準備をする
　　つり銭切れでなければ金銭を受け付ける
　投入された金銭の種類を判別する
　有効な金銭の場合
　　投入金額に加算する
　　投入金額を表示する
　　投入された金銭を保管する
　無効な金銭の場合
　　投入された金銭を返却する

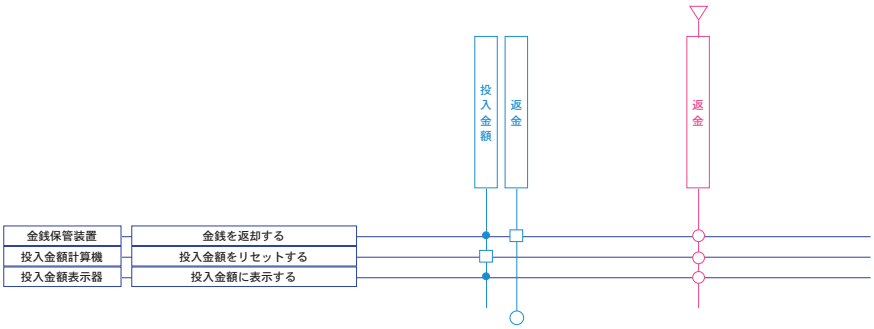


代金を差し引く
　投入金額から商品代金を差し引く
　投入金額を表示する



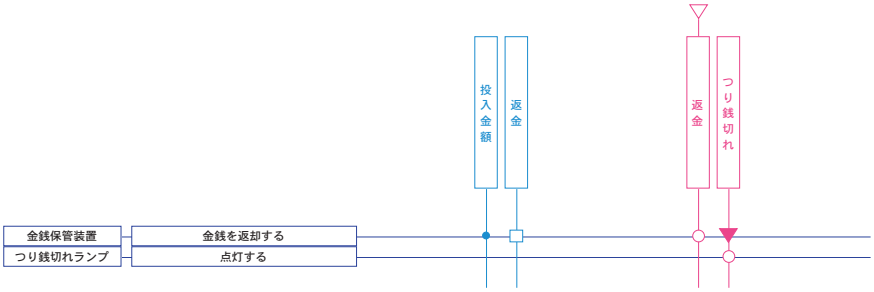
返金する
　投入金額を返金口に搬出する

　金額に応じた硬貨を返金口に搬出する
　投入金額＝０にする
　投入金額＝０を表示する



つり銭切れを表示する
　つり銭切れ時はつり銭切れランプを点灯する
　（※ランプ点灯時は金銭をを受け付けないので復帰はしない）

　返金後、保有硬貨数が規定数以下になったら
　つり銭切れランプを点灯する



–商品管理機構–

（缶ジュース販売機と変りありません）

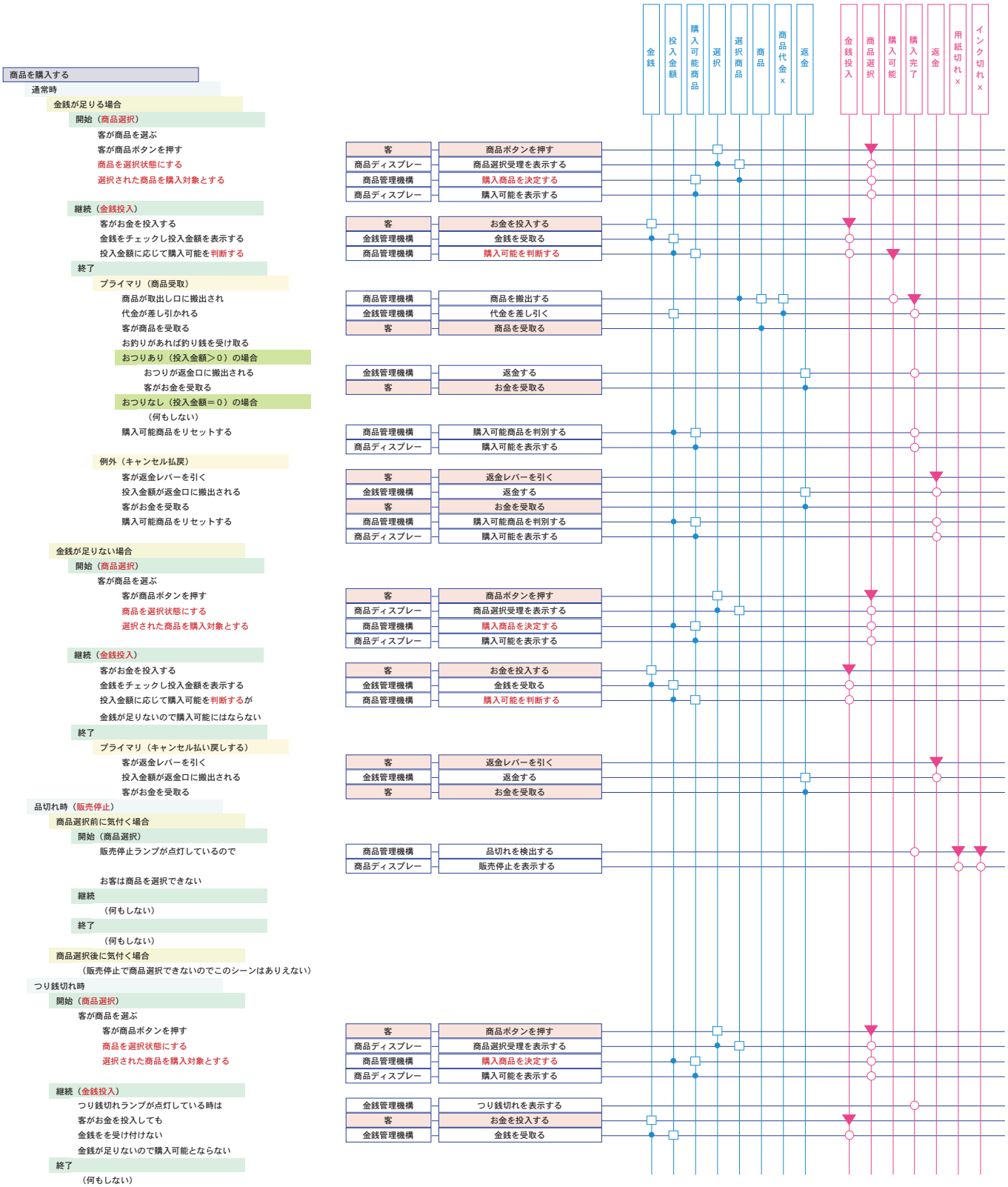
–商品ディスプレー–

（缶ジュース販売機と変りありません）

–商品棚–

（缶ジュース販売機と変りありません）

［券売機］–ルート–



商品を購入する

通常時

金銭が足りる場合

開始（商品選択）

客が商品を選ぶ

客が商品ボタンを押す

商品を選択状態にする

選択された商品を購入対象とする

継続（金銭投入）

客がお金を投入する

金銭をチェックし投入金額を表示する

投入金額に応じて購入可能を判断する

終了

プライマリ（商品受取）

商品が取出し口に搬出され

代金が差し引かれる

客が商品を受取る

お釣りがあれば釣り銭を受け取る

おつりあり（投入金額＞０）の場合

おつりが返金口に搬出される

客がお金を受取る

おつりなし（投入金額＝０）の場合

（何もしない）

購入可能商品をリセットする

例外（キャンセル払戻）

客が返金レバーを引く

投入金額が返金口に搬出される

客がお金を受取る

購入可能商品をリセットする

金銭が足りない場合

開始（商品選択）

客が商品を選ぶ

客が商品ボタンを押す

商品を選択状態にする

選択された商品を購入対象とする

継続（金銭投入）

客がお金を投入する

金銭をチェックし投入金額を表示する

投入金額に応じて購入可能を判断するが

金銭が足りないので購入可能にはならない

終了

プライマリ（キャンセル払い戻しする）

客が返金レバーを引く

投入金額が返金口に搬出される

客がお金を受取る

品切れ時（販売停止）

商品選択前に気付く場合

開始（商品選択）

販売停止ランプが点灯しているので

お客は商品を選択できない

継続

（何もしない）

終了

（何もしない）

商品選択後に気付く場合

（販売停止で商品選択できないのでこのシーンはありえない）

つり銭切れ時

開始（商品選択）

客が商品を選ぶ

客が商品ボタンを押す

商品を選択状態にする

選択された商品を購入対象とする

継続（金銭投入）

つり銭切れランプが点灯している時は

客がお金を投入しても

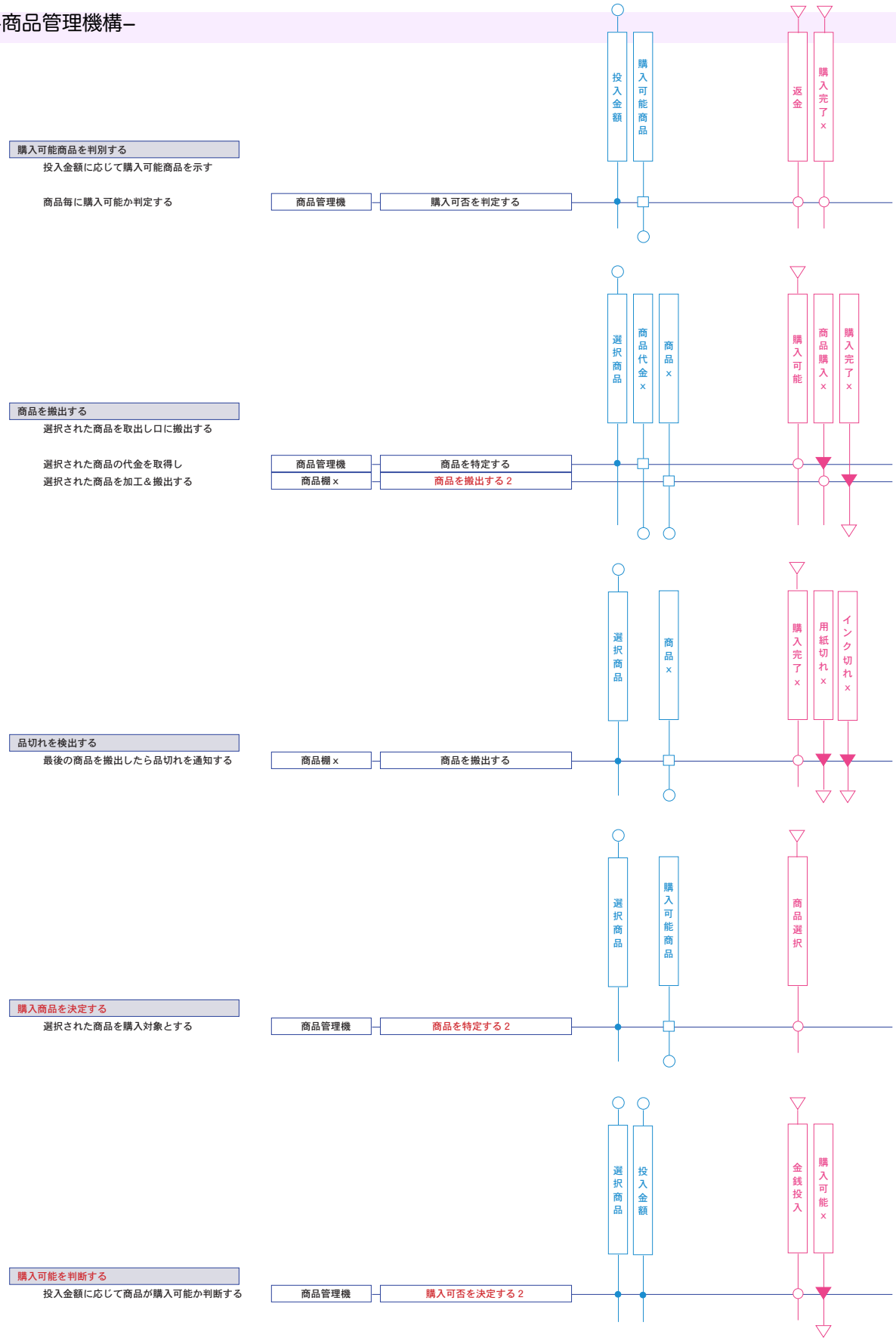
金銭をを受け付けない

金銭が足りないので購入可能とならない

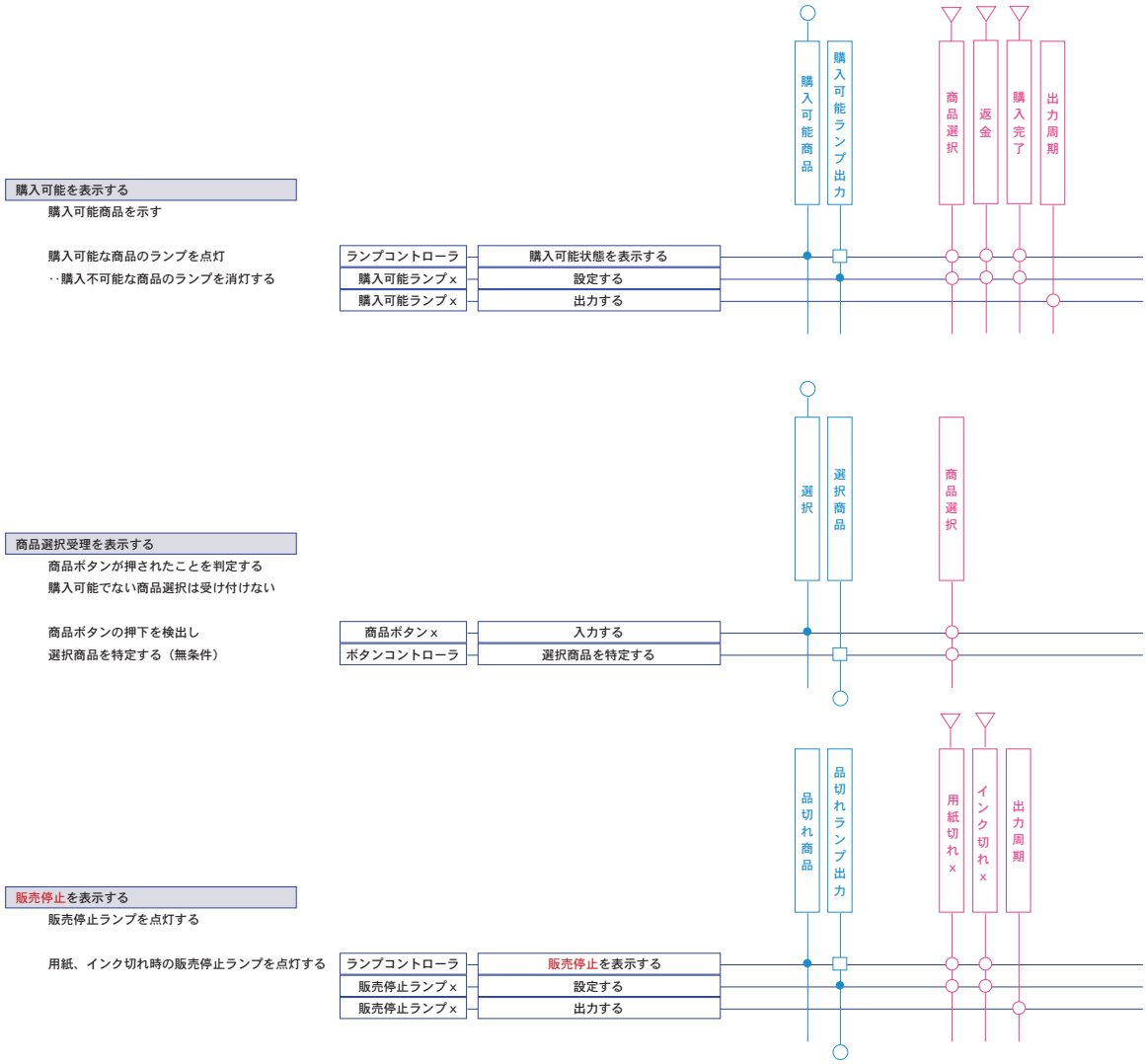
終了

（何もしない）

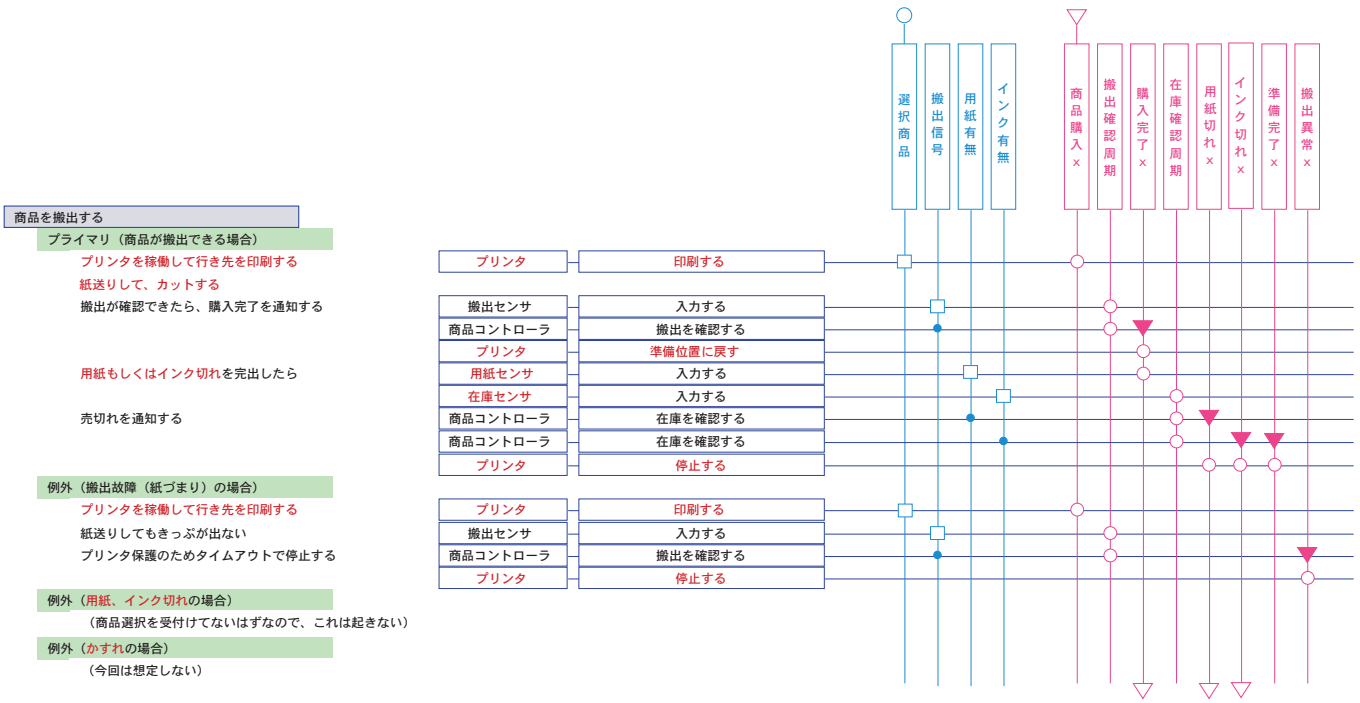
–商品管理機構–



–商品ディスプレー–



–商品棚–

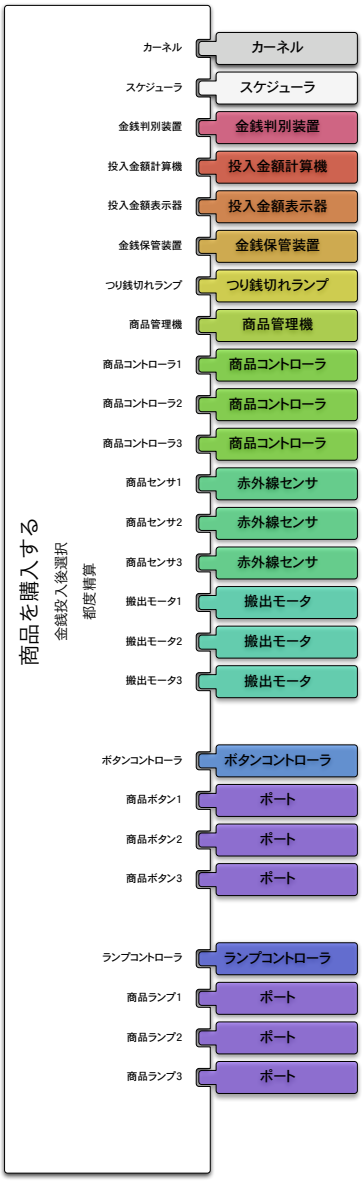


ソース

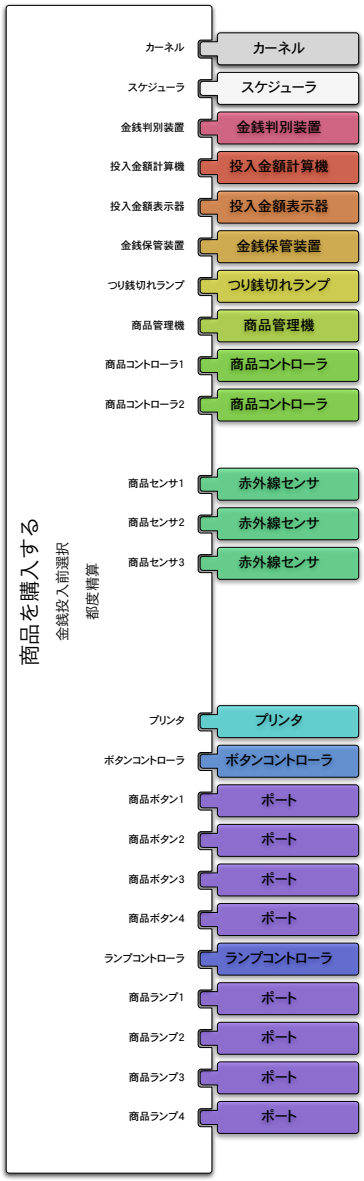
［実装］

缶ジュース販売機、券売機、たばこ販売機ともほとんどのコンポーネントは共通です
フレームワークとコンポーネントでバリエーションが表現できることが解ると思います

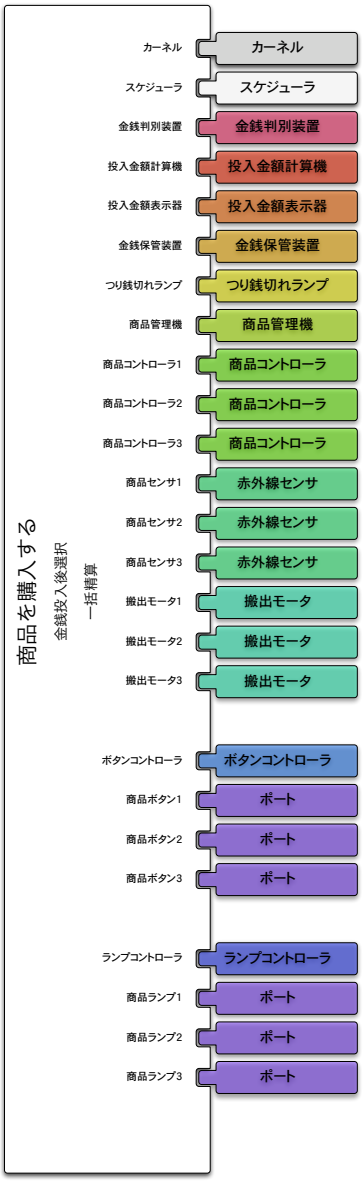
【缶ジュース販売機】



【券売機】



【たばこ販売機】



ー金銭判別装置ー

```
package {
	public class 金銭判別装置 {
//	インスタンス定数
		public var 金銭受理習性;
		public var 金銭拒否習性;
		public var 判別装置リセット;
		public var 十円センサ;
		public var 百円センサ;
		public var 千円センサ;
		public var 返却出力;
		public var ＯＮ;
//	インスタンス変数
		public var m_name;
		public var 金額;
//	ユニバーサルメソッド
		public var Get:Function;
		public var Set:Function;
		public var PostEvent:Function;
//	コンストラクタ
		public function 金銭判別装置 (a_Name, a_Arg:Array) {
			m_name = a_Name;
			金銭受理習性 = a_Arg[0];
			金銭拒否習性 = a_Arg[1];
			判別装置リセット = a_Arg[2];
			十円センサ = a_Arg[3];
			百円センサ = a_Arg[4];
			千円センサ = a_Arg[5];
			返却出力 = a_Arg[6];
			ＯＮ = a_Arg[7];
			金額= 0;
		}
//	クラスメソッド
		public function 金銭の種類を判別する () {
			if (Get(十円センサ) == ＯＮ)
				金額 = 10;
			else if (Get(百円センサ) == ＯＮ)
				金額 = 100;
			else if (Get(千円センサ) == ＯＮ)
				金額 = 1000;
			if (金額 >= 0)
				PostEvent(金銭受理習性, "B");
			else
				PostEvent(金銭拒否習性, "B");
		}
		public function リセットする () {
			Set(判別装置リセット, ＯＮ);
			金額 = 0;
		}
		public function 投入金銭を返却する () {
```

```
                Set(返却出力, ＯＮ);
            }
        }
}
```

ー投入金額計算機ー

```
package {
    public class 投入金額計算機 {
//      インスタンス定数
//      インスタンス変数
            public var m_name;
            public var 投入金額;
//      ユニバーサルメソッド
            public var Get:Function;
//      コンストラクタ
            public function 投入金額計算機 (a_Name:String, a_Arg:Array) {
                m_name = a_Name;
                投入金額 = 0;
            }
//      クラスメソッド
            public function 投入金額に加算する (a_金額:String) :void {
                投入金額 += Get(a_金額);
            }
            public function 投入金額から減算する (a_商品代金:String) :void {
                投入金額 -= Get(a_商品代金);
            }
            public function 投入金額をリセットする () :void {
                投入金額 = 0;
            }
    }
}
```

ー投入金額表示器ー

```
package {
    public class 投入金額表示器 {
//      インスタンス定数
            public var 表示ポート;
//      インスタンス変数
            public var m_name;
//      ユニバーサルメソッド
            public var Get:Function;
            public var Set:Function;
//      コンストラクタ
            public function 投入金額表示器 (a_Name:String, a_Arg:Array) {
                m_name = a_Name;
                表示ポート = a_Arg[0];
            }
//      クラスメソッド
            public function 投入金額を表示する (a_投入金額:String) :void {
                var l_投入金額:int = Get(a_投入金額);
```

```
                Set(表示ポート, l_投入金額);
            }
        }
}
```

ー商品コントローラー

```
package {
    public class 商品コントローラ {
//      インスタンス定数
            public var 購入完了習性;
            public var 売切れ習性;
            public var 準備完了習性;
            public var 搬出異常習性;
            public var 搬出判定値:int;
            public var 準備判定値:int;
            public var 在庫判定値:int;
            public var タイムアウト判定値:int;
            public var 準備中;
            public var 搬出済;
            public var 売切れ;
            public var 異常;
//      インスタンス変数
            public var m_name;
            public var 状態;
            public var タイムアウトカウンタ:int;
//      ユニバーサルメソッド
            public var Get:Function;
            public var PostEvent:Function;
//      コンストラクタ
            public function 商品コントローラ (a_Name:String, a_Arg:Array) {
                m_name = a_Name;
                購入完了習性 = a_Arg[0];
                売切れ習性 = a_Arg[1];
                準備完了習性 = a_Arg[2];
                搬出異常習性 = a_Arg[3];
                搬出判定値 = a_Arg[4];
                準備判定値 = a_Arg[5];
                在庫判定値 = a_Arg[6];
                タイムアウト判定値 = a_Arg[7];
                準備中 = a_Arg[8];
                搬出済 = a_Arg[9];
                売切れ = a_Arg[10];
                異常 = a_Arg[11];
                状態 = 準備中;
                タイムアウトカウンタ = 0;
            }
//      クラスメソッド
            public function 搬出を確認する (a_搬出信号:String) :void {
                var l_搬出信号:int = Get(a_搬出信号);
                if (状態 == 準備中) {
```

```
                        if  (l_搬出信号 >= 搬出判定値) {
                                状態 = 搬出済;
                                タイムアウトカウンタ = 0;
                                PostEvent(購入完了習性, "B");
                        }
                        else {
                                if (タイムアウトカウンタ > 0) {
                                        タイムアウトカウンタ--;
                                        if (タイムアウトカウンタ == 0) {
                                                状態 = 異常;
                                                PostEvent(搬出異常習性, "B");
                                        }
                                }
                        }
                }
        }
        public function 在庫を確認する (a_在庫有無:String) :void {
                var l_在庫有無:int = Get(a_在庫有無);
                if (状態 == 搬出済) {
                        if (l_在庫有無 <= 在庫判定値) {
                                状態 = 売切れ;
                                タイムアウトカウンタ = 0;
                                PostEvent(売切れ習性, "B");
                        }
                        else if (l_在庫有無 <= 準備判定値) {
                                状態 = 準備中;
                                タイムアウトカウンタ = 0;
                                PostEvent(準備完了習性, "B");
                        }
                        else {
                                if (タイムアウトカウンタ > 0) {
                                        タイムアウトカウンタ--;
                                        if (タイムアウトカウンタ == 0) {
                                                状態 = 異常;
                                                PostEvent(搬出異常習性, "B");
                                        }
                                }
                        }
                }
        }
        public function リセットする () :void {
                タイムアウトカウンタ = タイムアウト判定値;
        }
    }
}
```

ー赤外線センサー

```
package {
	public class 赤外線センサ {
//	インスタンス定数
		public var ポート;
//	インスタンス変数
		public var m_name;
		public var バッファ:int;
//	ユニバーサルメソッド
		public var Get:Function;
//	コンストラクタ
		public function 赤外線センサ (a_Name:String, a_Arg:Array) {
			m_name = a_Name;
			ポート = a_Arg[0];
			バッファ = 0;
		}
//	クラスメソッド
		public function 入力する () :void {
			バッファ = Get(ポート);
		}
	}
}
```

ー搬出モーターー

```
package {
	public class 搬出モータ {
//	インスタンス定数
		public var ポート;
		public var 停止;
		public var 順転;
		public var 逆転;
//	インスタンス変数
		public var m_name;
		public var 駆動状態;
//	ユニバーサルメソッド
		public var Set:Function;
//	コンストラクタ
		public function 搬出モータ (a_Name:String, a_Arg:Array) {
			m_name = a_Name;
			ポート = a_Arg[0];
			停止 = a_Arg[1];
			順転 = a_Arg[2];
			逆転 = a_Arg[3];
			駆動状態 = 0;
		}
//	クラスメソッド
		public function 準備側に回転する () :void {
			駆動状態 = 逆転;
			Set(ポート,駆動状態);
		}
		public function 搬出側に回転する () :void {
```

```
                駆動状態 = 順転;
                Set(ポート,駆動状態);
        }
        public function 停止する () :void {
                駆動状態 = 停止;
                Set(ポート,駆動状態);
        }
    }
}
```

ープリンターー

```
package {
    public class プリンタ {
//      インスタンス定数
        public var ポート;
        public var バッファ;
        public var 停止;
        public var 印字;
        public var 給紙;
//      インスタンス変数
        public var m_name;
        public var 駆動状態;
//      ユニバーサルメソッド
        public var Get:Function;
        public var Set:Function;
//      コンストラクタ
        public function プリンタ (a_Name:String, a_Arg:Array) {
                m_name = a_Name;
                ポート = a_Arg[0];
                バッファ = a_Arg[1];
                停止 = a_Arg[2];
                印字 = a_Arg[3];
                給紙 = a_Arg[4];
                駆動状態 = 0;
        }
//      クラスメソッド
        public function 準備する () :void {
                Set(バッファ, "");
                駆動状態 = 給紙;
                Set(ポート, 駆動状態);
        }
        public function 印刷する (a_印刷データ) :void {
                var l_印刷データ = Get(a_印刷データ);
                Set(バッファ, l_印刷データ);
                駆動状態 = 印字;
                Set(ポート, 駆動状態);
        }
        public function 停止する () :void {
                駆動状態 = 停止;
                Set(ポート, 駆動状態);
```

```
		}
	}
}
```

ーボタンコントローラー

```
package {
	public class ボタンコントローラ {
//	インスタンス定数
		public var ボタン配置;
		public var ＯＮ;
//	インスタンス変数
		public var m_name;
		public var 選択商品;
//	ユニバーサルメソッド
		public var Get:Function;
//	コンストラクタ
		public function ボタンコントローラ (a_Name:String, a_Arg:Array) {
			m_name = a_Name;
			ＯＮ = a_Arg[0];
			ボタン配置 = new Object();
			for (i=1; i<a_Arg.length; i+=2) {
				ボタン配置[a_Arg[i]] = a_Arg[i+1];
			}
			選択商品 = "";
		}
//	クラスメソッド
		public function 選択商品を特定する (a_購入可能商品:String) :void {
		//	選択されているボタンを調べる
			選択商品 = "";
			var n = 0;
			for (var name in ボタン配置) {
				var l_ボタン = Get(ボタン配置[name]);
				if (l_ボタン == ＯＮ) {
					選択商品 = name;
					n++;
				}
			}
		//	複数選択は無効
			if (n > 1) {
				選択商品 = "";
				return;
			}
		//	購入可能商品の指定がなければ商品選択を確定する
			if (a_購入可能商品 == "ＡＬＬ")
				return;
		//	購入可能商品の指定があれば該当する商品選択を確定する
			else {
				var l_購入可能商品０ = Get(a_購入可能商品);
				l_購入可能商品 = l_購入可能商品０.toString().split(",");
				for (var i:int=0; i<l_購入可能商品.length; i++) {
```

```
                        if (l_購入可能商品[i] == 選択商品)
                            break;
                    }
                    if (i >= l_購入可能商品.length)
                        選択商品 = "";
                }
            }
        }
}
```

ーランプコントローラー

```
package {
    public class ランプコントローラ {
//      インスタンス定数
            public var 消灯;
            public var 購入可能点灯;
            public var 品切れ点灯;
//      インスタンス変数
            public var m_name;
            public var ランプ配置;
//      ユニバーサルメソッド
            public var Get:Function;
//      コンストラクタ
            public function ランプコントローラ (a_Name:String, a_Arg:Array) {
                m_name = a_Name;
                消灯 = a_Arg[0];
                購入可能点灯 = a_Arg[1];
                品切れ点灯= a_Arg[2];
                ランプ配置 = new Object();
                for (i=3; i<a_Arg.length; i++) {
                    ランプ配置[a_Arg[i]] = 消灯;
                }
            }
//      クラスメソッド
            public function 購入可能状態を表示する (a_購入可能商品:String) :void {
                var l_購入可能商品０ = Get(a_購入可能商品);
                var l_購入可能商品 = l_購入可能商品０.toString().split(",");
                //  一旦消灯する
                for (var name in ランプ配置) {
                    if (ランプ配置[name] != 品切れ点灯)
                        ランプ配置[name] = 消灯;
                }
                //  購入可能商品に該当するランプを検索する
                for (var i:int=0; i<l_購入可能商品.length; i++) {
                    //  品切れでなかったら購入可能にする
                    if (ランプ配置[l_購入可能商品[i]] != 品切れ点灯)
                        ランプ配置[l_購入可能商品[i]] = 購入可能点灯;
                }
            }
            public function 品切れを表示する (a_品切れ商品:String) :void {
```

```
                //   品切れ商品に該当するランプを品切れ点灯にする
                ランプ配置[a_品切れ商品] = 品切れ点灯;
            }
        }
}
```

ーポートー

```
package {
        public class ポート {
//      インスタンス定数
                public var ポートレジスタ;
//      インスタンス変数
                public var m_name;
                public var バッファ;
//      ユニバーサルメソッド
                public var Get:Function;
                public var Set:Function;
//      コンストラクタ
                public function ポート (a_Name:String, a_Arg:Array) {
                        m_name = a_Name;
                        ポートレジスタ = a_Arg[0];
                }
//      クラスメソッド
                public function 入力する () :void {
                        バッファ = Get(ポートレジスタ);
                }
                public function データを設定する (a_値:String) :void {
                        var l_値 = Get(a_値);
                        if (l_値 != null)
                                バッファ = l_値;
                }
                public function 出力する () :void {
                        Set(ポートレジスタ, バッファ);
                }
        }
}
```