**詳細設計書**

**自動販売機シミュレータ**

**開発プロジェクト**

株式会社メイテック IT05Aグループ

池田、臼井、小野寺、小林

**更新履歴**

|  |  |  |  |
| --- | --- | --- | --- |
| バージョン | 更新日付 | 更新者 | 内容 |
| 1.0 | 2022/06/22 | 臼井 | 新規作成 |
| 1.1 | 2022/06/23 | 小野寺 | 図・データベースの修正、  機能詳細の修正 |
| 1.2 | 2022/06/23 | 小野寺 | 更新履歴の修正 |
| 1.3 | 2022/06/23 | 臼井 | シーケンス図の修正 |
| 1.4 | 2022/06/23 | 小野寺 | フィードバック後の修正 |

**目次**

[**クラス図** 3](#_Toc106703791)

[**アクティビティ図** 4](#_Toc106703792)

[**シーケンス図** 5](#_Toc106703793)

[**モジュール構成図** 6](#_Toc106703794)

[**機能処理詳細** 7](#_Toc106703795)

[**データベース定義(詳細)** 10](#_Toc106703796)

# **クラス図**

VendingMachine

+ 使用可能紙幣硬貨確認()

+ 投入金額の合計()

+ データベース更新()

+ 投入金額上限確認()

+ 総売上高管理()

+ 投入金額表示()

+ 購入可能商品判定表示と売り切れ表示()

+ 釣り切れ表示()

+ お釣り計算()

私たちが作成する自動販売機シミュレータは、MainクラスとPurchaserクラス、VendingMachineクラスの3つのクラスに分かれる。

ただし、Mainクラスは、PurchaserクラスやVendingMachineクラスのインスタンス生成のために用いられるので、メソッドやフィールドを含まない。

そのため、Mainクラスのクラス図は存在しない。

また、VendingMachineクラスはフィールドを含まない。

以上を踏まえて、クラス図を以下に示す。

1

Purchaser

+ money (投入された金額)

+ button (購入者が入力したボタン番号)

+ 金額投入

+ ボタン選択

1

VendingMachine

+ 使用可能紙幣硬貨確認( )

+ 投入金額の合計( )

+ データベース更新( )

+ 投入金額上限確認( )

+ 総売上高管理( )

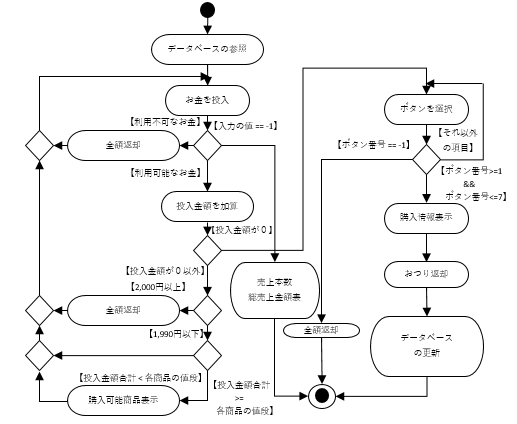
+ 購入可能商品判定表示と売り切れ表示( )

+ 釣り切れ表示( )

+ お釣り計算( )

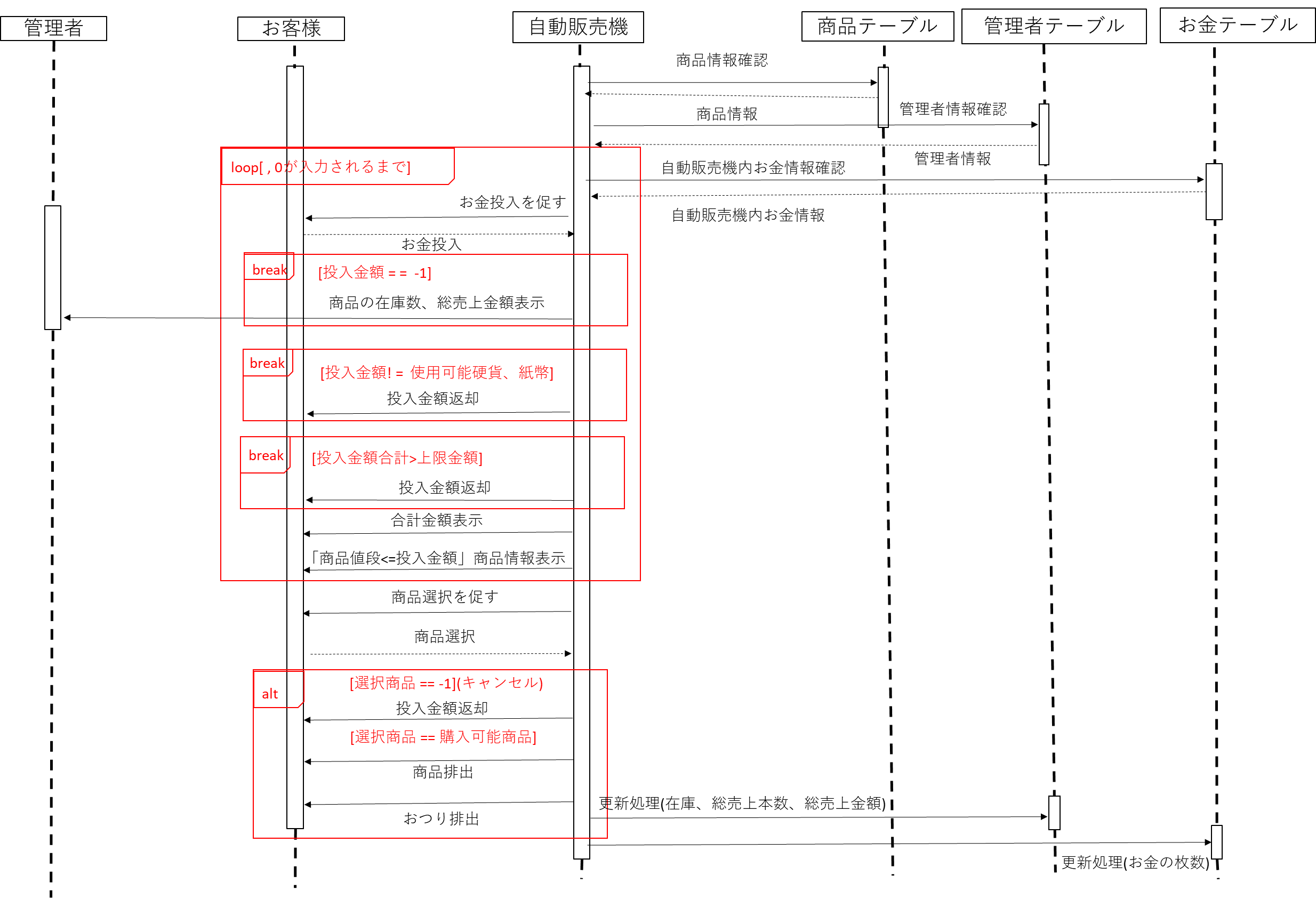
# **アクティビティ図**

私たちが作成する自動販売機のシミュレータの全体の流れを表したアクティビティ図を以下に示す。



# **シーケンス図**

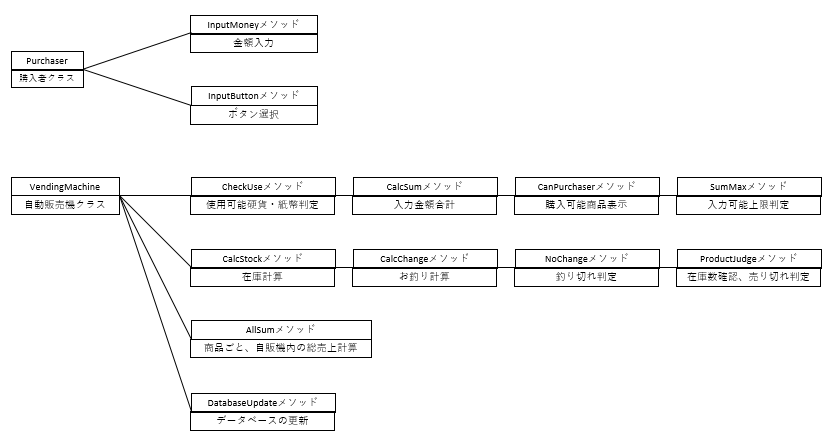
私たちが作成する自動販売機シミュレータのシーケンス図を以下に示す。



# **モジュール構成図**

各クラスのメソッドの処理内容を簡潔に図式化した。

私たちが作成する自動販売機シミュレータのモジュール構成図を以下に示す。



# **機能処理詳細**

Purchaserクラスは投入金額の入力、ボタンの入力を主に行う。

釣銭や使用可能な紙幣・硬貨の判定等の処理はVendingMachineクラスで行うものとする。

Mainクラス内では初めに、データベースで定義した商品のテーブル「product」と自動販売機内にあるお金のテーブル「money」を配列に全て保存し、その配列を用いて処理を行う。

処理を行い、データベースの内容を更新する必要がある場合は、データベースの更新を行う。

それぞれのクラスのフィールドやメソッド、引数や戻り値等の機能処理の詳細を以下に示す。

ただし、VendingMachineクラスはフィールドを定義しない。

**Purchaserクラス**

|  |  |  |
| --- | --- | --- |
| 変数名 | データ型 | 変数内容 |
| money | int | 入力された金額の値を保持する。 |
| button | int | 入力されたボタン番号の値を保持する。 |

|  |  |  |  |
| --- | --- | --- | --- |
| 関数名 | 戻り値 | 引数 | 詳細処理内容 |
| Input | int | void | 「お金を投入してください」と金額投入を促す。  Scanner関数を用いて金額の入力を行う。  戻り値として、入力した金額を返す。 |
| InputButton | int | void | 「ボタンを入力してください」とボタン番号の選択を促す。  Scanner関数を用いて商品ボタンを入力する。  -1が入力された場合は返却ボタンが押されたとし、全額返金する。  -1、1～7以外の入力された場合、再度入力を促す。  戻り値として、選択したボタン番号を返す。 |

**VendingMachineクラス**

|  |  |  |  |
| --- | --- | --- | --- |
| 関数名 | 戻り値 | 引数 | 詳細処理内容 |
| CheckUse | boolean | int money | 引数で受け取った入力金額に対して、  1,000、500、100、50、10、0以外の数値が  入力された場合は、使用不可能な紙幣・硬貨が  含まれているとして一度全額返金する。 |
| CalcNum | int | int money, int NUM[ ] | PurchaserクラスのInput関数で入力された  金額とテーブル名「money」の要素の  自動販売機内にある各金額の枚数NUM[ ]を  引数に用いる。  戻り値として、投入された合計金額を返す。 |
| DatabaseUpdate | void | void | データベースの更新を行う。 |
| ProductJudge | boolean[ ] | int STOCK[ ] | 各商品の在庫数を確認する。  テーブル名「manage」の要素の各商品の在庫数STOCK[ ]を引数に用いる。  戻り値として、各商品の在庫数が0の場合はTRUE、それ以外の場合はFALSEを返す。 |
| SumMax | boolean | int sum | 投入金額の上限確認を行う。  投入金額の合計を引数に用いる。  戻り値として、投入金額の合計が上限の  1,990円を超えた場合はTRUE、それ以外の  場合はFALSEを返す。 |
| AllSum | int | int SUM[ ] | 自動販売機の総売上金額を計算する。  テーブル名「manage」の要素の各商品の売上高であるSUMを引数に用いる。  戻り値として、自動販売機内の総売上金額を  返す。 |
| CanPurchase | void | int sum,  String NAME[ ],  int VALUE[ ],  int STOCK[ ],  int BUTTON[ ] | 投入金額に対して、購入可能な商品の表示、  売り切れの表示、投入された合計金額を  表示する。  投入金額合計sum、テーブル名「product」の  要素の商品NAME、値段VALUE、テーブル名「manage」の要素の在庫数STOCK、ボタン番号BUTTONを引数に用いる。戻り値はない。 |

|  |  |  |  |
| --- | --- | --- | --- |
| NoChange | boolean[ ] | int NUM[ ] | 自動販売機内にあるお金が釣り切れかどうかを判定する。テーブル名「money」の要素の  自動販売機内にある各金額の枚数NUM[ ]を  引数に用いる。戻り値として、各金額の枚数が  4枚以下の場合はTRUE、それ以外の場合はFALSEを返す。 |
| CanChange | boolean | int sum  int NUM[]  int VALUE[]  int button | ある商品が購入された場合に自動販売機内にあるお金でお釣りを払えるかを判定する。  投入金額合計sum、テーブル名「money」の要素である自動販売機内にある各金額の枚数NUM[ ]、テーブル名「product」の要素の値段VALUE[ ]、入力されたボタン番号buttonを引数に用いる。投入金額合計から入力されたボタン番号に対応する商品の値段分を減算する。  残金を自動販売機内の硬貨をなるべく少なく  なるように組み合わせ、各硬貨の枚数を計算し、自動販売機内のお金で払える場合TRUE、払えない場合FALSEを返す。 |
| CalcChange | int [ ] | int sum,  int NUM[ ]  int VALUE[ ]  int button | お釣りの計算を行う。投入金額合計のsum、  テーブル名「money」の要素である自動販売機内にある各金額の枚数NUM[ ]、テーブル名「product」の要素の値段VALUE[ ]、入力されたボタン番号buttonを引数に用いる。投入金額  合計から入力されたボタン番号に対応する商品の値段分を減算する。  残金を自動販売機内の硬貨をなるべく少なく  なるように組み合わせ、各硬貨の枚数を戻り値とする。 |
| CalcStock | int [] | int STOCK[ ],  int button | 選択されたボタンに対応する在庫数を減らす。  入力されたボタン番号button、テーブル名「manage」の要素の在庫数STOCK[ ]を引数に  用いる。入力されたボタン番号に対応する商品の在庫数を1本減らす。各商品の在庫数を戻り値とする。 |

# **データベース定義(詳細)**

商品の詳細を示すproductというテーブルと管理者が扱うmanageというテーブル、自動販売機内にある釣銭の残高を示すmoneyというテーブルの3つのテーブルが存在する。

テーブル名：product

|  |  |  |
| --- | --- | --- |
| 項目 | データ型 | 定義 |
| 商品ID(ID) | VARCHAR (5) | 商品識別番号。商品毎に異なる。 |
| 商品名(NAME) | VARCHAR (20) | 商品の名前。 |
| 値段(VALUE) | INT | 商品の値段。  110円、120円、130円を必ず含む。 |
| ボタン番号(BUTTON) | TINYINT | 各商品に対応するボタン番号。  ただし、同一商品に複数ボタン番号が  存在する場合もある。 |

テーブル名：manage

|  |  |  |
| --- | --- | --- |
| 項目 | データ型 | 定義 |
| ボタン番号(BUTTON) | TINYINT | 各商品に対応するボタン番号。  ただし、同一商品に複数ボタン番号が  存在する場合もある。 |
| 総売上本数(SUMBOTTLE) | INT | ボタン毎の総売上本数。 |
| 総売上金額(SUM) | INT | ボタン毎の総売上金額。 |
| 在庫数(STOCK) | TINYINT | ボタン毎の在庫数。 |

テーブル名：money

|  |  |  |
| --- | --- | --- |
| 項目 | データ型 | 定義 |
| 金額(MONEY) | INT | 硬貨や紙幣の金額。 |
| 枚数(NUM) | TINYINT | 各硬貨や紙幣の枚数。 |

**自動販売機データ一覧**

|  |  |  |  |
| --- | --- | --- | --- |
| 商品ID | 商品名 | 値段 | ボタン番号 |
| 00001 | Cola | 130 | 1 |
| 2 |
| 00002 | GreenTea | 110 | 3 |
| 00003 | Water | 100 | 4 |
| 00004 | Coffee | 110 | 5 |
| 00005 | EnergyDrink | 120 | 6 |
| 00006 | Yagult10000 | 150 | 7 |

<備考>

* ボタン番号1と2は同一商品(Cola)が格納されている。
* 100円から150円の商品が取り揃えられている。
* 商品は6種類用意されている。
* ボタンは7つある。
* 下線部を主キーとし、主キーはボタン番号である。
* 商品IDは商品名とリンクしている。

**管理者テーブルデータ一覧**

|  |  |  |  |
| --- | --- | --- | --- |
| ボタン番号 | 総売上本数 | 総売上金額 | 在庫数 |
| 1 | 0 | 0 | 20 |
| 2 | 0 | 0 | 20 |
| 3 | 0 | 0 | 20 |
| 4 | 0 | 0 | 20 |
| 5 | 0 | 0 | 20 |
| 6 | 0 | 0 | 20 |
| 7 | 0 | 0 | 20 |

<備考>

* 自動販売機の在庫上限数は20本であり、在庫数の初期状態は各商品20本ずつである。
* 総売り上げ本数の初期状態は全商品0本である。
* ボタン番号に関しては、自動販売機のデータ一覧と同様である。

**釣銭テーブルデータ一覧**

|  |  |
| --- | --- |
| 硬貨・紙幣名 | 枚数 |
| 1,000 | 30 |
| 500 | 30 |
| 100 | 30 |
| 50 | 30 |
| 10 | 30 |

<備考>

* 自動販売機内の釣銭上限数は50枚であり、各釣銭の初期状態は30枚ずつである。
* 使用可能な硬貨や紙幣は「10円、50円、100円、500円、1,000円」の5種類である。