Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

*Факультет программной инженерии и компьютерной техники*

Теория систем

Лабораторная работа №1

Группа: P3324

Выполнил:

Круглов Егор Ильич

Преподаватель:

Русак Алёна Викторовна

Санкт-Петербург

2025г.

# Задание

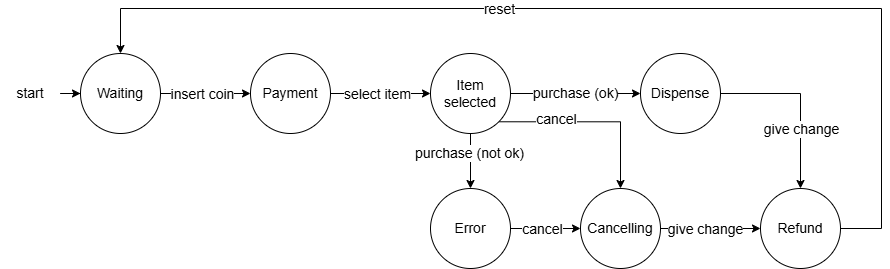
Требуется спроектировать управляющий конечный автомат.

# Решение

В качестве конечного автомата был выбран торговый автомат. Опишем его состояния:

* Waiting – ожидание того, что с ним начнут взаимодействовать (вносить деньги)
* Payment – состояние внесения денег в автомат
* Item selected – состояние, когда пользователь выбрал товар
* Dispense – состояние выдачи товара
* Refund – состояние возврата денег (сдачи или при отмене)
* Error – состояние ошибки
* Cancelling – состояние отмены

Диаграмма состояний:



Описание конечного автомата:

Состояния (X): waiting, payment, item\_selected, dispense, error, cancelling, refund.

Входные сигналы (U): insert\_coin, select\_item, confirm\_purchase, cancel, give\_change, reset.

Выходные сигналы (Y): waiting, payment, item\_selected, dispense, error, cancelling, refund.

Функция перехода задается таблицей:

|  |  |  |  |
| --- | --- | --- | --- |
| **Триггер** | **Исходное состояние** | **Целевое состояние** | **Условия/действия** |
| insert\_coin | waiting, payment | payment | Пополнение баланса |
| select\_item | payment | item\_selected | Выбор товара |
| confirm\_purchase | item\_selected | dispense | Успешная покупка (\_buy\_item) |
| confirm\_purchase | item\_selected | error | Ошибка (\_show\_error) |
| canel | item\_selected, error | cancelling | Отмена, подготовка к возврату |
| give\_change | dispense, cancelling | refund | Возврат остатка |
| reset | refund | waiting | Возвращение в начальное состояние (\_reset) |

Код представлен на GitHub - <https://github.com/KruglovEgor/System-Theory/tree/main/lab1>

!pip install transitions

**from** **transitions** **import** Machine

**class** **VendingMachine**(object):

**def** **\_\_init\_\_**(self):

self.balance = **0**

self.selected\_item = None

self.items = {

'A1': {'name': 'Вода', 'price': **50**},

'A2': {'name': 'Шоколад', 'price': **70**}

}

**def** **\_buy\_item**(self):

**if** self.selected\_item **in** self.items **and** self.balance >= self.items[self.selected\_item]['price']:

**print**(f"Покупаем товар {self.selected\_item} за {self.items[self.selected\_item]['price']}.")

self.balance -= self.items[self.selected\_item]['price']

**return** True

**return** False

**def** **\_show\_error**(self):

**if** self.selected\_item **not** **in** self.items:

**print**("Нет такого товара!")

**elif** self.balance < self.items[self.selected\_item]['price']:

**print**("Недостаточно денег!")

**else**:

**print**("Какая-то неизвестная ошибка :(")

**def** **\_reset**(self):

**if** self.balance > **0**:

**print**(f"Возвращено денег: {self.balance}")

self.balance = **0**

self.selected\_item = None

**def** **\_insert\_coin**(self, amount): !

self.balance += amount

**print**(f"Баланс пополнен на {amount}. Текущий баланс: {self.balance}")

**def** **\_select\_item**(self, item):

self.selected\_item = item

**print**(f"Выбран товар: {item}.")

# Инициализация автомата

machine = VendingMachine()

# Определение состояний и переходов

states = ['waiting', 'payment', 'item\_selected', 'dispense', 'refund', 'error', 'cancelling']

transitions = [

{

'trigger': 'insert\_coin',

'source': ['waiting', 'payment'],

'dest': 'payment',

'after': '\_insert\_coin'

},

{

'trigger': 'select\_item',

'source': 'payment',

'dest': 'item\_selected',

'after': '\_select\_item'},

{

'trigger': 'confirm\_purchase',

'source': 'item\_selected',

'dest': 'dispense',

'conditions': '\_buy\_item',

'after': 'give\_change'

},

{

'trigger': 'confirm\_purchase',

'source': 'item\_selected',

'dest': 'error',

'unless': '\_buy\_item',

'after': '\_show\_error'

},

{

'trigger': 'cancel',

'source': ['item\_selected', 'error'],

'dest': 'cancelling',

'after': 'give\_change'

},

{

'trigger': 'give\_change',

'source': ['dispense', 'cancelling'],

'dest': 'refund',

'after': 'reset'

},

{'trigger': 'reset',

'source': 'refund',

'dest': 'waiting',

'after': '\_reset'}

]

# Создаем конечный автомат

fsm = Machine(model=machine, states=states, transitions=transitions, initial='waiting')

# Пример использования

**print**("--- Успешная покупка ---")

machine.insert\_coin(**50**)

machine.select\_item('A1')

machine.confirm\_purchase()

**print**(f"Текущее состояние: {machine.state}")

**print**("**\n**--- Сценарий ошибки ---")

machine.insert\_coin(**30**)

machine.select\_item('A2')

machine.confirm\_purchase()

**print**(f"Текущее состояние: {machine.state}")

**print**("**\n**--- Возврат денег ---")

machine.cancel()

**print**(f"Текущее состояние: {machine.state}")

# Вывод

В ходе лабораторной работы был спроектирован и реализован конечный автомат, имитирующий работу торгового аппарата.

1. Конечный автомат был описан:
   * Описание на языке множеств X, U, Y
   * Представлена таблица переходов
   * Построена диаграмма состояний
2. Был написан код на языке Python:
   * Корректная обработка 7 состояний и 6 переходов.
   * Интеграция проверок баланса и валидации товаров.
3. Особенности реализации:
   * Разделение логики на методы-колбэки (\_buy\_item, \_show\_error).
   * Использование условий (conditions) для ветвления переходов.
4. Результаты тестирования подтвердили:
   * Успешную выдачу товара при достаточном балансе.
   * Блокировку операций при ошибках.
   * Работоспособность механизма возврата денег.