Задание №15
Простое моделирование процессов

1. Общая постановка задачи

Выполните задание, соответствующее вашему варианту.

Текст программы снабдите подробными комментариями.

Опишите набор примеров, демонстрирующих все аспекты работы вашей программы.

Опишите программу в текстовом файле с именем `task15-NN.rb`, где `NN` — номер вашего варианта. Полученный файл загрузите на портал в качестве выполненного задания.

2. Предварительные замечания

Наличие в решении указанных в задании классов, их элементов и методов обязательно. Решение может быть засчитано только если задание выполнено на 100%.

Кроме перечисленных в задании элементов разрешается вводить в решение любое количество дополнительных классов и методов на усмотрение студента, при условии, что все добавленные элементы служат для решения поставленной задачи, а не для расширения функциональности реализованного класса.

В решении не следует использовать конструкции цикла `while` и `until`.

Не следует делать предположений на счет задания, не сформулированных явно в условии. Если возникают сомнения — задайте вопрос на форуме «Язык Ruby».

3. Пример выполнения задания

ТЕКСТ ЗАДАЧИ: Музыкальный автомат содержит некоторое количество музыкальных треков, которые он может проигрывать.

Реализуйте класс `MusicMachine`, моделирующий работу музыкального автомата. В классе должна быть определена константа `MaxTrackCount` — максимальное количество треков, которое может быть загружено в автомат. При инициализации автомат может получать хеш, содержащий названия треков и время длительности каждого из них (некоторое число). Двух треков с одинаковым названием в автомат загрузить нельзя.

Должны быть реализованы следующие методы:

- `empty` — опустошить автомат: удалить все треки из него. Метод возвращает сам автомат.
- `add_track(name, time)` — добавить в автомат трек с именем `name` и временем проигрывания `time`. Трек добавляется, если трек с таким именем в автомате отсутствовал. Метод возвращает сам автомат.
- `play names` — поставить в очередь для проигрывания список треков, определенный массивом имен треков `names`. Если указанное в списке наименование отсутствует в автомате, то это имя игнорируется. Метод возвращает сам автомат.
- `pass` — проиграть первый трек в очереди для проигрывания (убрать его из головы очереди). Метод возвращает сам автомат.
- `count_time_to_play` — посчитать суммарное время треков, оставшихся в очереди для проигрывания.

РЕШЕНИЕ: Содержимое файла `task15-NN.rb`:

```ruby
class MusicMachine
  def initialize(hs = {})
    @track_count = 0  # количество треков в машине
    @tracks = {}      # хеш треков
    @play_queue = []  # очередь наименований треков для проигрывания
    # инициализация множества треков
    hs.keys.each { |key| add_track(key, hs[key]) }
  end

  MaxTrackCount = 15

```

```ruby
  def play names
    names.each do |name|
      if @tracks[name]           # если есть в базе трек с наименованием
        @play_queue.push(name)   # то его можно проиграть
      end
    end
    self
  end

  def pass
    @play_queue.shift
    self
  end

  def count_time_to_play
    # суммируем в аккумулятор все длительности треков в очереди
    @play_queue.inject(0) { |acc, name| acc + @tracks[name] }
  end

  def add_track(name, time)
    # если максимальное количество треков не превышено
    # и трек с таким названием отсутствует в базе
    if @track_count < MaxTrackCount and not @tracks[name]
        @tracks[name] = time    # добавляем трек в хеш
        @track_count += 1
    end
    self
  end

  def empty
    @track_count = 0
    @tracks = {}
    @play_queue = []
    self
  end
end

##
## ПРИМЕР ВЫЗОВА
##
# Создаем новый автомат с тремя треками
a = MusicMachine.new({"my 1" => 17.6, "my 2" => 13, "my 3" => 25})
# Добавляем еще два трека, после чего заказываем проиграть два
a.add_track("new 1", 11).add_track("new 2", 7).play(["new 2", "my 1"])
# Заказываем еще один
a.play(["new 1"])
# Просим показать, сколько времени осталось до конца проигрывания
puts a.count_time_to_play
# Заказываем еще три (из которых два не существуют)
a.play(["new 3", "new 4", "my 1", "my 3"])
# Просим показать, сколько времени осталось до конца проигрывания
puts a.count_time_to_play
# Пропустим один трек
a.pass
puts a.count_time_to_play
a.pass
puts a.count_time_to_play
# Очистим автомат, после чего загрузим в него два трека
# и попросим проиграть два
a.empty.add_track("new 1", 11).add_track("new 2", 7).play(["new 2", "my 1"])
puts a.count_time_to_play
```

Файл с примером можно загрузить с портала.

4. Варианты заданий

1. В кофейном аппарате 6 емкостей: для кофе, чая, какао, сливок, лимона и сахара. Объем каждой емкости задается порциями. Максимальный объем каждой емкости рассчитан на одинаковое количество порций. Аппарат может готовить напитки по заданному набору рецептов. Рецепт представляет собой набор ингредиентов из имеющихся, с указанием количества порций каждого из них. Пользователь аппарата может при желании в заказе увеличить или уменьшить количество каждого из ингредиентов рецепта на одну порцию, добавить одну порцию лимона (если лимон не входит в рецепт) и добавить от одной до девяти порций сахара. Аппарат готовит напиток, только если все заказанные ингредиенты есть в наличии.

Опишите класс `CoffeeMachine`, моделирующий работу кофейного аппарата. В классе должен быть описан хеш-константа `Recipes`, содержащий рецепты напитков для автоматов данного класса, и константа — объем емкостей для аппаратов данного класса. Ключом в `Recipes` является строка — название напитка, а каждым значением — хеш с названиями и количеством ингредиентов. Названия ингредиентов, используемые в программе — строки `"coffee"`, `"tea"`, `"chocolate"`, `"cream"`, `"lemon"`, `"sugar"`. Количество порций в каждом рецепте — целое число. Должны быть определены следующие методы:

- `load` — заполнить на 100% все емкости аппарата. Результат — загруженный аппарат.
- `order(rec, changes)` — заказать напиток, где `rec` — строка, наименование рецепта, а `changes` — хеш, содержащий необходимые изменения в рецепте. Для любого изменения задается целое число — количество порций, которое нужно увеличить или уменьшить вхождение ингредиента в рецепт. Если указанный ингредиент отсутствует в рецепте (за исключением сахара и лимона), то он игнорируется. Если для ингредиента, отличного от сахара, указано изменение больше `1`, то оно принимается равным `1` (для сахара величина не может быть больше `9`, в противном случае принимается `9`). Если для ингредиента указано изменение меньше чем `-1`, то оно принимается равным `-1`. Метод должен изменять состояние аппарата и выдавать `true`, если напиток успешно приготовлен или `nil`, если напиток приготовить нельзя.
- `status` — определить состояние аппарата. Метод должен возвращать хеш, описывающий оставшиеся ингредиенты.
- `stat` — выдать статистику со времени последней загрузки аппарата. Метод должен возвращать хеш, описывающий количество заказов по каждому рецепту после последней заправки аппарата ингредиентами (для рецептов, по которым были заказы).

2 (Бонус 10%)**.** Банкомат выдает и принимает купюры определенного достоинства. Предполагается, что в банкомате есть по две кассеты для каждого номинала купюр: одна кассета с купюрами для выдачи и вторая — для принятых купюр. Все кассеты имеют одну и ту же емкость для всех банкоматов. Банкомат может выдавать купюры только из кассет для выдачи и принимать купюры только в кассеты для приема. Если кассета для приема заполнена, то банкомат перестает принимать купюры этого достоинства.

Опишите класс `CashMachine`, моделирующую работу банкомата. В классе должен быть описан константа-массив `Banknotes` номиналов купюр, с которыми могут работать банкоматы заданного класса, и константа — объем одной кассеты банкомата. При инициализации банкомата должна быть задана максимальная сумма, которую можно снять в банкомате за одну операцию. Должны быть определены следующие методы:

- `load` — заполнить на 100% все кассеты для выдачи купюр. Результат — заполненный банкомат.
- `empty` — опустошить на 100% все кассеты для приема купюр. Результат — опустошенный банкомат.
- `new_max n` — установить максимально возможную сумму для снятия в банкомате. Результат — сам банкомат.
- `deposit hs` — внести деньги в банкомат. `hs` — хеш, сопоставляющий номиналам вносимых купюр их количество. Метод должен изменять состояние банкомата и возвращать массив, первый элемент которого — принятая сумма, второй элемент — хеш, описывающий непринятые купюры.
- `withdraw s` — снять сумму `s`, где `s` — целочисленное значение. Метод должен изменять состояние банкомата и выдавать хеш, описывающий выданные купюры. В случае, если банкомат не может выдать указанную сумму, метод должен возвращать `nil`.
- `status` — определить состояние банкомата. Метод должен возвращать массив, первый элемент которого — сумма в оставшихся для выдачи купюрах в банкомате, второй — хеш, описывающий оставшиеся для выдачи купюры, третий — сумма принятых купюр, четвертый — хеш, описывающий принятые купюры, находящиеся в банкомате.

3 (Бонус 10%)**.** В автомате для торговли напитками (в бутылках) имеется несколько торговых рядов для товара. В каждом ряду может быть выставлен один вид товара. Одно и то же наименование может быть выставлено в нескольких рядах автомата. Каждый вид товара имеет свое наименование. Покупатель вносит сумму-депозит (в купюрах), после чего может выбрать по наименованию товар для покупки. Если в депозит внесена достаточная сумма, то автомат выдает товар, снимая сумму с депозита. Покупатель может запросить возврат не израсходованного депозита. Автомат выдает ту сумму, которую он способен вернуть в номиналах имеющихся купюр.

Предполагается, что в автомате есть кассеты для каждого вида купюр, которые могут пополняться при внесении депозита. Излишки купюр из кассет поступают в общую емкость с денежной массой, из которой деньги извлекает только обслуживающий персонал (деньги из емкости обратно в кассеты не поступают).

Реализуйте класс `BottleMachine`, моделирующий работу автомата. В классе должны определяться константы `RowCount` и `PlaceNumber` — количество рядов товара и количество единиц товара в каждом ряду соответственно для всех автоматов данного класса. Кроме того, должен быть описан константа-массив `Banknotes` номиналов купюр, с которыми могут работать автоматы заданного класса, и константа — объем одной кассеты для купюр. Должны быть определены следующие методы:

- `empty` — опустошить автомат: извлекается весь товар и все деньги. Результат выполнения метода — опустошенный автомат.
- `load hs` — загрузить автомат товаром, где `hs` — хеш, в котором для каждого наименования товара указывается его цена и количество. Если товар уже присутствует в незаполненном ряду автомата, то загрузка товара в автомат начинается с дозаполнения этого ряда (и продолжается в свободных рядах). Если товар присутствовал в автомате, но имел другую цену — цена товара обновляется. Результат выполнения метода — хеш, описывающий оставшийся товар, не поместившийся в автомате.
- `deposit hs` — внести деньги в автомат. `hs` — хеш, сопоставляющий номиналам вносимых купюр их количество. Метод должен изменять состояние автомата. Результат выполнения метода — сам автомат.
- `order name` — заказать товар, где `name` — строка, наименование товара. Если товар присутствует в автомате и депозит содержит сумму, достаточную для его покупки, то списывается его цена с депозита и товар выдается. Результат метода — сам автомат.
- `withdraw` — вернуть депозит. Автомат выдает не израсходованный депозит теми купюрами, которые есть у него в наличии. Если невозможно выдать сумму депозита, то выдается максимально возможная сумма, а оставшаяся сумма остается в депозите. Результат выполнения метода — хеш, описывающий выданные купюры.
- `status` — определить состояние автомата. Метод должен возвращать массив, первый элемент которого — хеш, описывающий оставшийся в автомате товар, второй — денежная сумма, находящаяся в автомате, третий — хеш, описывающий банкноты в автомате.
- `stat` — выдать статистику со времени последнего опустошения аппарата. Метод должен возвращать хеш, содержащий для каждого наименования проданного товара сумму, полученную от его продажи, и количество проданных единиц.

4 (Бонус 10%)**.** В автомате по продаже газировки несколько емкостей для разных сиропов (концентратов). Объем каждой емкости задается порциями. Максимальный объем каждой емкости рассчитан на одинаковое количество порций. Автомат может выдавать газировку без сиропа, с сиропом или с двойным сиропом.

Покупатель вносит сумму-депозит (в купюрах), после чего может выбрать напиток для покупки. Если в депозит внесена достаточная сумма и в автомате есть выбранный сироп, то автомат выдает напиток, снимая сумму с депозита. Покупатель может запросить возврат не израсходованного депозита. Автомат выдает ту сумму, которую он способен вернуть в номиналах имеющихся купюр. Предполагается, что в автомате есть кассеты для каждого вида купюр, которые могут пополняться при внесении депозита. Излишки купюр из кассет поступают в общую емкость с денежной массой, из которой деньги извлекает только обслуживающий персонал (деньги из емкости обратно в кассеты не поступают).

Опишите класс `SodaMachine`, моделирующий работу автомата. В классе должны быть описаны константы — объем емкостей для аппаратов данного класса и максимальное количество емкостей. Кроме того, должен быть описан константа-массив `Banknotes` номиналов купюр, с которыми могут работать автоматы заданного класса, и константа — объем одной кассеты для купюр. Названия сиропов, используемые в программе, — строки. Количество порций — целое число. При инициализации автомата должны указываться количество емкостей в автомате и стоимость воды без сиропа. Должны быть определены следующие методы:

- `load hs` — заполнить аппарат. Здесь `hs` — хеш, описывающий загружаемые сиропы: для каждого наименования указывается предложенный объем, стоимость за порцию. Если в автомате есть емкость с таким сиропом, то она дозаполняется. Иначе заполняется первая свободная емкость, пока все емкости не будут заполнены или не кончится предлагаемый сироп. Если стоимость порции сиропа в автомате имела другую стоимость, то цена в автомате обновляется. Результат — хеш, описывающий наименования и количество незагруженного сиропа.
- `empty` — опустошить на 100% все емкости автомата. Результат — опустошенный автомат.
- `set_tanks n` — опустошить на 100% все емкости автомата и оставить в автомате `n` пустых емкостей. Результат — опустошенный автомат.
- `water_price` — установить стоимость воды без сиропа.
- `deposit hs` — внести деньги в автомат. `hs` — хеш, сопоставляющий номиналам вносимых купюр их количество. Метод должен изменять состояние автомата. Результат выполнения метода — сам автомат.
- `order(name, n)` — заказать товар, где `name` — строка, наименование сиропа (если отсутствует, то выдается вода без сиропа), а `n` — может принимать значение 1 или 2 (1 по умолчанию) — одинарный или двойной

сироп. Если сироп присутствует в автомате и депозит содержит сумму, достаточную для его покупки, то списывается его цена с депозита и товар выдается. В противном случае ничего не происходит. Результат метода — сам автомат.

- `withdraw` — вернуть депозит. Автомат выдает не израсходованный депозит теми купюрами, которые есть у него в кассетах. Если невозможно выдать сумму депозита, то выдается максимально возможная сумма, а оставшаяся сумма остается в депозите. Результат выполнения метода — хеш, описывающий выданные купюры.
- `status` — определить состояние автомата. Метод должен возвращать массив, первый элемент которого — хеш, описывающий оставшийся набор сиропов в автомате, второй— денежная сумма, находящаяся в автомате, третий — хеш, описывающий банкноты в автомате.
- `stat` — выдать статистику со времени последнего опустошения аппарата. Метод должен возвращать хеш, содержащий для каждого наименования проданного сиропа сумму, полученную от его продажи, и количество проданных порций.

5 (Бонус 20%)**.** Автомат для размена денег выдает и принимает купюры определенного достоинства. Предполагается, что в автомате есть несколько кассет для купюр. Все кассеты имеют одну и ту же емкость для всех автоматов данного класса. Кассета может быть использована для любого номинала купюр, но после того, как в него помещена купюра определенного достоинства, в него можно поместить купюры только этого достоинства (до тех пор, пока кассета не опустеет). Автомат принимает сумму для размена в имеющиеся кассеты и выдает ее с разменом (в случае, если это возможно). Одни и те же кассеты служат для приемы и выдачи купюр.

Опишите класс `CashMachine`, моделирующую работу автомата. В классе должен быть описан константа-массив `Banknotes` номиналов купюр, с которыми могут работать автоматы заданного класса, и константа — объем одной кассеты автомата. При инициализации автомата должно быть задано количество кассет в автомате и максимальная сумма, по которой может производиться размен. Должны быть определены следующие методы:

- `load hs` — заполнить автомат заданным набором банкнот. Здесь `hs` — хеш, задающий для каждого номинала банкнот их количество. Если банкноты определенного номинала уже присутствуют в автомате и кассета с ними не заполнена, то заполнение начинается с этой кассеты. Если кассета с определенным номиналом заполнена полностью, то начинается заполнение первой свободной кассеты. Результат метода — хеш, описывающий незагруженный набор купюр.
- `set_cassette n` — опустошить на 100% все кассеты автомата и оставить в автомате `n` пустых кассет. Результат — опустошенный автомат.
- `empty` — опустошить на 100% все кассеты автомата. Результат — опустошенный автомат.
- `new_max n` — установить максимально возможную сумму для размена. Результат — сам автомат.
- `change hs` — разменять деньги. `hs` — хеш, сопоставляющий номиналам вносимых купюр их количество. Метод должен изменять состояние автомата и возвращать хеш, описывающий возвращаемые купюры. Автомат принимает купюры и пытается выдать принятую сумму купюрами, достоинством меньше минимальной из внесенных. Если это сделать невозможно, то предполагается размен, с минимально возможным максимальным номиналом купюр. Если невозможно принять купюры в автомат (нет свободного места в кассетах для размещения), то сумма возвращается обратно без размена.
- `status` — определить состояние автомата. Метод должен возвращать массив, первый элемент которого — сумма в автомате, второй — хеш, описывающий купюры в автомате, третий — массив, описывающий кассеты в автомате: по каждой кассете — номинал и количество банкнот в ней.