Author: María Paulina García Velásquez
Repo: [María Paulina García / coffee-machine · GitLab](#)

Problem:

- Public coffee machine at a university

Functional requirements:

- The machine must prepare and serve coffee
- The machine must make several types of coffee: cappuccino, espresso, mocha, water chocolate, milk chocolate, vanilla cappuccino and decaffeinated.
- The coffee must have the right amount of sugar.
- The machine must accept a payment and return a remainder.
- The machine must show all available options.
- An operator must be able to collect money and refill the machine (powder for each type of coffee, water, sugar)
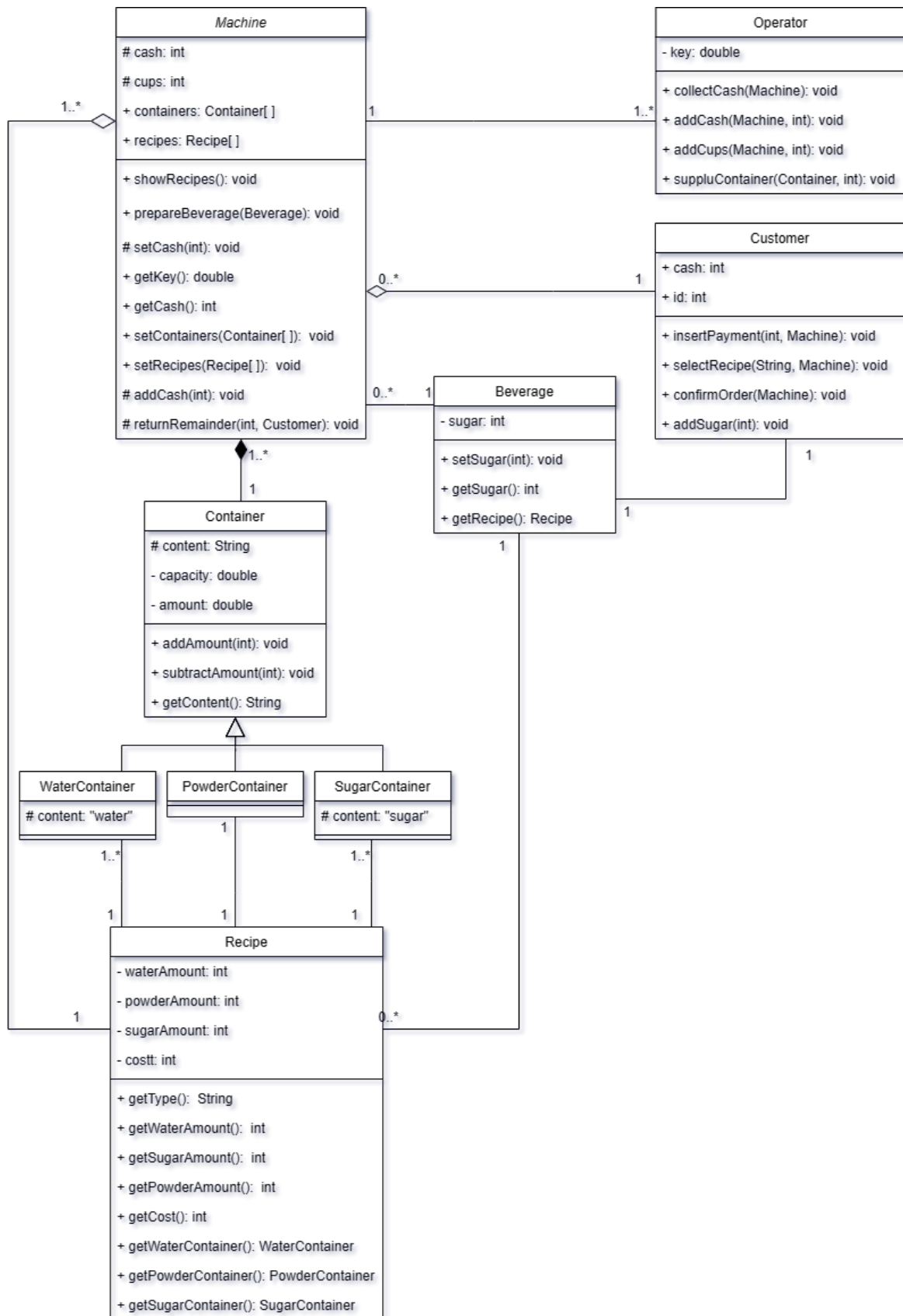
Non-functional requirements:

- The machine must make the coffee in a fast and efficient way.
- The machine must consume a low amount of energy.
- The machine must have the right amount of water and recipe mixtures available.
- The programme must not use up a lot of memory (machines have low amount of memory available)

CRC cards

Machine	
Responsibilties	**Collaborators**
Show the recipes	Operator
Prepare beverages	Customer
Return the remainder cash to the customer	Beverage
	Container
	Recipe

Operator	
Responsibilties	**Collaborators**
Collect the cash of the machine	Machine
Add cash to the machine	
Add cups to the machine	
Supply the containers	

Customer	
Responsibilties	**Collaborators**
Insert payment into the machine	Machine
Select the recipe	Beverage
Add sugar	
Confirm order	

Recipe	
Responsibilties	**Collaborators**
	Container
	Machine
	Beverage

Beverage	
Responsibilties	**Collaborators**
	Machine
	Recipe
	Customer

Container	
Responsibilties	**Collaborators**
Add content	Machine
Substract content	Recipe

UML Class diagram

Machine

- # cash: int
- # cups: int
- + containers: Container[ ]
- + recipes: Recipe[ ]

- + showRecipes(): void
- + prepareBeverage(Beverage): void
- # setCash(int): void
- + getKey(): double
- + getCash(): int
- + setContainers(Container[ ]): void
- + setRecipes(Recipe[ ]): void
- # addCash(int): void
- # returnRemainder(int, Customer): void

Operator

- - key: double

- + collectCash(Machine): void
- + addCash(Machine, int): void
- + addCups(Machine, int): void
- + suppluContainer(Container, int): void

Customer

- + cash: int
- + id: int

- + insertPayment(int, Machine): void
- + selectRecipe(String, Machine): void
- + confirmOrder(Machine): void
- + addSugar(int): void

Beverage

- - sugar: int

- + setSugar(int): void
- + getSugar(): int
- + getRecipe(): Recipe

Container

- # content: String
- - capacity: double
- - amount: double

- + addAmount(int): void
- + subtractAmount(int): void
- + getContent(): String

WaterContainer

- # content: "water"

PowderContainer

SugarContainer

- # content: "sugar"

Recipe

- - waterAmount: int
- - powderAmount: int
- - sugarAmount: int
- - costt: int

- + getType(): String
- + getWaterAmount(): int
- + getSugarAmount(): int
- + getPowderAmount(): int
- + getCost(): int
- + getWaterContainer(): WaterContainer
- + getPowderContainer(): PowderContainer
- + getSugarContainer(): SugarContainer

Use case diagram

Coffe vending machine app

Operator

Customer

Machine

Supply materials
<< includes>> Add cash
<< includes>> Add cups

Collect cash
<< includes>> Supply container

Select recipe
<< extends>> Show recepies

<< extends>> Order drink
<< includes>> Add sugar

<< extends>> Confirm order
<< extends>> Prepare coffee

Insert payment
<< extends>> Calculate cost

<< extends>>

<< includes>> Return remainder