Arcade architecture

Generated by Doxygen 1.10.0

Chapter 1

Commit norm :

<Gitmoji> : [Element / Module] : [MESSAGE]

Gitmoji = The emoji approriate for the current modification. [Element / Module] = The elemenet you applied the modification. [MESSAGE] = A detail message of what you did.

Gitmojies:

```
Code feature :
    - :sparkles: (): Introduce new features
    - :recycle: (): Refactor / update code
    - :bug: (): Fix a bug
    - :poop: () : Remove Coding style or temporary fix
    - :rotating_light: () : Fix Compiling Warning
    - :fire: (): Remove code or files

Test feature :
    - :white_check_mark: (): Add, update, or pass tests

Architecture :
    - :see_no_evil: (): Add or update .gitignore files
    - :construction_worker: (): Add or update CI build system
    - :building_construction: () : Make Architectural changes
    - :memo: () : Add or update documentation

...
```

1.0.1 Pull Request

- :tada: (): This Gitmoji must be used for each PR created!
- :lipstick: (): This Gitmoji must be used for each PR merged!
- :rewind: (): This Gitmoji must be used for each revert done!

1.1 Git-Cli :

- Changer message de commit, avant qu'il soit push :
  ```
  git commit --amend -m "New commit message"
  ```
- Changer le message de commit, si il a deja été push :
  ```
  git commit --amend -m "New commit message"
  git push --force
  ```
- Un-add un ficher add par erreur qui est pas encore push:
  ```
  git restore --staged <file>
  ```
- Un-add un fichier qui a été commit :
  ```
  git reset --soft HEAD~1
  git restore --staged fichier-a-retirer.txt
  git commit -m "Nouveau message de commit (sans le fichier)"
  ```

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

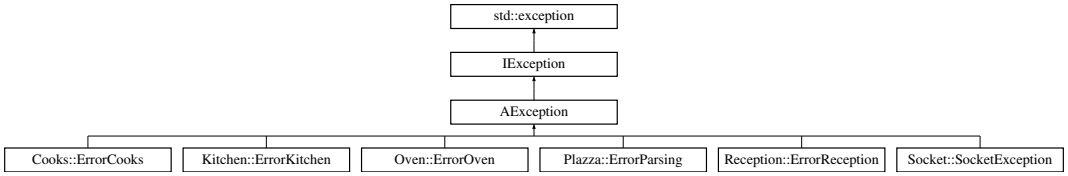
Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 AException Class Reference

Inheritance diagram for AException:



Public Member Functions

- **AException** (const std::string &type, const std::string &message)
- const char * what () const noexcept override
- std::string getType () const noexcept override
- std::string getMessage () const noexcept override
- std::string getFormattedMessage () const noexcept override

Private Attributes

- std::string **_message**
- std::string **_type**

5.1.1 Member Function Documentation

5.1.1.1 getFormattedMessage()

```
std::string AException::getFormattedMessage ( ) const  [inline], [override], [virtual], [noexcept]
```

Implements IException.

5.1.1.2 getMessage()

```
std::string AException::getMessage ( ) const  [inline], [override], [virtual], [noexcept]
```

Implements IException.

5.1.1.3 getType()

```
std::string AException::getType ( ) const  [inline], [override], [virtual], [noexcept]
```

Implements IException.

5.1.1.4 what()

```
const char * AException::what ( ) const  [inline], [override], [virtual], [noexcept]
```

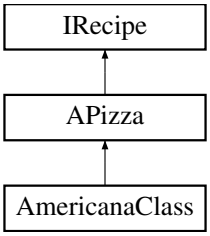
Implements IException.

The documentation for this class was generated from the following file:

- common/AException.hpp

5.2 AmericanaClass Class Reference

Inheritance diagram for AmericanaClass:



Public Member Functions

- **AmericanaClass** (int number)
- void cook (int cookTime) override
- std::shared_ptr< Ingridient > prepare (int number, std::shared_ptr< Ingridient > ingridient) override
- void serve () override

Public Member Functions inherited from APizza

- **APizza** (int number)
- int getNumber () const override
- void setNumber (int number) override

5.2.1 Member Function Documentation

5.2.1.1 cook()

```
void AmericanaClass::cook (
            int cookTime )  [override], [virtual]
```

Implements APizza.

5.2.1.2 prepare()

```
std::shared_ptr< Ingridient > AmericanaClass::prepare (
            int number,
            std::shared_ptr< Ingridient > ingridient )  [override], [virtual]
```

Implements APizza.

5.2.1.3 serve()

```
void AmericanaClass::serve ( )  [override], [virtual]
```

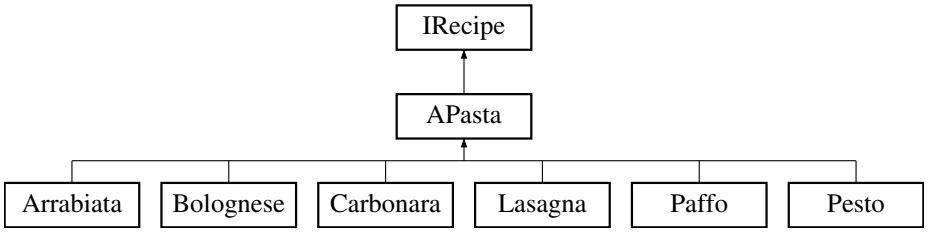
Implements APizza.

The documentation for this class was generated from the following files:

- src/recipes/pizza/Americana.hpp
- src/recipes/pizza/Americana.cpp

5.3 APasta Class Reference

Inheritance diagram for APasta:



Public Member Functions

- **APasta** (int number)
- virtual void cook (int cookTime) override=0
- virtual std::shared_ptr< Ingridient > prepare (int number, std::shared_ptr< Ingridient > ingridient) override=0
- virtual void serve () override=0
- int getNumber () const override
- void setNumber (int number) override

Private Attributes

- int **_size**
- int **_number**

5.3.1 Member Function Documentation

5.3.1.1 cook()

```
virtual void APasta::cook (
            int cookTime )  [override], [pure virtual]
```

Implements IRecipe.

5.3.1.2 getNumber()

```
int APasta::getNumber ( ) const  [override], [virtual]
```

Implements IRecipe.

5.3.1.3 prepare()

```
virtual std::shared_ptr< Ingridient > APasta::prepare (
            int number,
            std::shared_ptr< Ingridient > ingridient )  [override], [pure virtual]
```

Implements IRecipe.

5.3.1.4 serve()

```
virtual void APasta::serve ( )  [override], [pure virtual]
```

Implements IRecipe.

5.3.1.5 setNumber()

```
void APasta::setNumber (
            int number )  [override], [virtual]
```

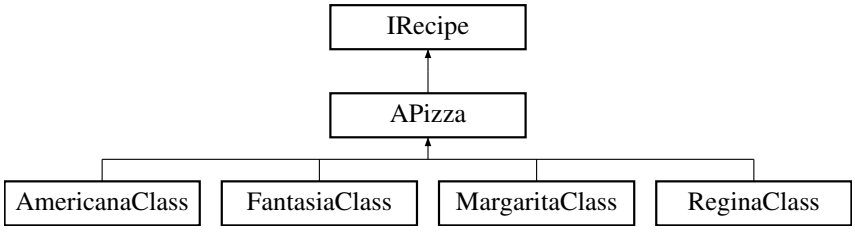
Implements IRecipe.

The documentation for this class was generated from the following files:

- common/APasta.hpp
- common/APasta.cpp

5.4 APizza Class Reference

Inheritance diagram for APizza:



Public Member Functions

- **APizza** (int number)
- virtual void cook (int cookTime) override=0
- virtual std::shared_ptr< Ingridient > prepare (int number, std::shared_ptr< Ingridient > ingridient) override=0
- virtual void serve () override=0
- int getNumber () const override
- void setNumber (int number) override

Private Attributes

- int **_number**

5.4.1 Member Function Documentation

5.4.1.1 cook()

```
virtual void APizza::cook (
            int cookTime )  [override], [pure virtual]
```

Implements IRecipe.

5.4.1.2 getNumber()

```
int APizza::getNumber ( ) const  [override], [virtual]
```

Implements IRecipe.

5.4.1.3 prepare()

```
virtual std::shared_ptr< Ingridient > APizza::prepare (
            int number,
            std::shared_ptr< Ingridient > ingridient )  [override], [pure virtual]
```

Implements IRecipe.

5.4.1.4 serve()

```
virtual void APizza::serve ( )  [override], [pure virtual]
```

Implements IRecipe.

5.4.1.5 setNumber()

```
void APizza::setNumber (
            int number )  [override], [virtual]
```

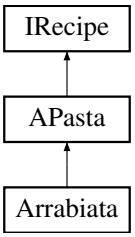
Implements IRecipe.

The documentation for this class was generated from the following files:

- common/APizza.hpp
- common/APizza.cpp

5.5 Arrabiata Class Reference

Inheritance diagram for Arrabiata:



Public Member Functions

- **Arrabiata** (int number)
- void cook (int cookTime) override
- std::shared_ptr< Ingridient > prepare (int number, std::shared_ptr< Ingridient > ingridient) override
- void serve () override

Public Member Functions inherited from APasta

- **APasta** (int number)
- int getNumber () const override
- void setNumber (int number) override

5.5.1 Member Function Documentation

5.5.1.1 cook()

```
void Arrabiata::cook (
            int cookTime )  [override], [virtual]
```

Implements APasta.

5.5.1.2 prepare()

```
std::shared_ptr< Ingridient > Arrabiata::prepare (
            int number,
            std::shared_ptr< Ingridient > ingridient )  [override], [virtual]
```

Implements APasta.

5.5.1.3 serve()

```
void Arrabiata::serve ( )  [override], [virtual]
```

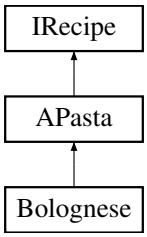
Implements APasta.

The documentation for this class was generated from the following files:

- src/recipes/pasta/Arrabiata.hpp
- src/recipes/pasta/Arrabiata.cpp

5.6 Bolognese Class Reference

Inheritance diagram for Bolognese:



Public Member Functions

- **Bolognese** (int number)
- void cook (int cookTime)
- std::shared_ptr< Ingridient > prepare (int number, std::shared_ptr< Ingridient > ingridient)
- void serve ()

Public Member Functions inherited from APasta

- **APasta** (int number)
- int getNumber () const override
- void setNumber (int number) override

5.6.1 Member Function Documentation

5.6.1.1 cook()

```
void Bolognese::cook (
            int cookTime )  [virtual]
```

Implements APasta.

5.6.1.2 prepare()

```
std::shared_ptr< Ingridient > Bolognese::prepare (
            int number,
            std::shared_ptr< Ingridient > ingridient )  [virtual]
```

Implements APasta.

5.6.1.3 serve()

```
void Bolognese::serve ( )  [virtual]
```

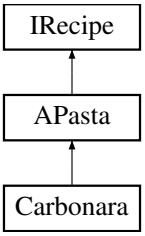
Implements APasta.

The documentation for this class was generated from the following files:

- src/recipes/pasta/Bolognese.hpp
- src/recipes/pasta/Bolognese.cpp

5.7 Carbonara Class Reference

Inheritance diagram for Carbonara:



Public Member Functions

- **Carbonara** (int number)
- void cook (int cookTime) override
- std::shared_ptr< Ingridient > prepare (int number, std::shared_ptr< Ingridient > ingridient) override
- void serve () override

Public Member Functions inherited from APasta

- **APasta** (int number)
- int getNumber () const override
- void setNumber (int number) override

5.7.1 Member Function Documentation

5.7.1.1 cook()

```
void Carbonara::cook (
            int cookTime )  [override], [virtual]
```

Implements APasta.

5.7.1.2 prepare()

```
std::shared_ptr< Ingridient > Carbonara::prepare (
            int number,
            std::shared_ptr< Ingridient > ingridient )  [override], [virtual]
```

Implements APasta.

5.7.1.3 serve()

```
void Carbonara::serve ( )  [override], [virtual]
```

Implements APasta.

The documentation for this class was generated from the following files:

- src/recipes/pasta/Carbonara.hpp
- src/recipes/pasta/Carbonara.cpp

5.8 Cooks Class Reference

Classes

- class ErrorCooks

Public Member Functions

- **Cooks** (std::shared_ptr< Ingridient > ingridient, int id, int cookTime, int restockTime)
- std::shared_ptr< Ingridient > **startOrder** (std::shared_ptr< Ingridient > ingridient, std::vector< std::string > order)
- void **restock** ()
- bool **hasEnoughIngredients** (const std::string &orderData, std::shared_ptr< Ingridient > ingridient)
- int **getID** () const
- bool **isBusy** () const
- bool **isRestocking** () const

Private Attributes

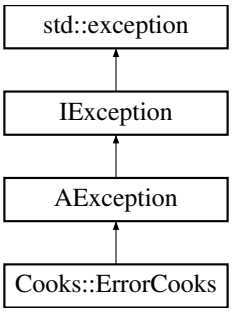
- int **_ID**
- int **_cookTime**
- int **_restockTime**
- std::shared_ptr< Ingridient > **_ingridient**
- std::mutex **_statusMutex**
- bool **_isBusy**
- bool **_isRestocking**

The documentation for this class was generated from the following files:

- src/cooks/Cooks.hpp
- src/cooks/Cooks.cpp

5.9 Cooks::ErrorCooks Class Reference

Inheritance diagram for Cooks::ErrorCooks:



Public Member Functions

- **ErrorCooks** (const std::string &message)

Public Member Functions inherited from AException

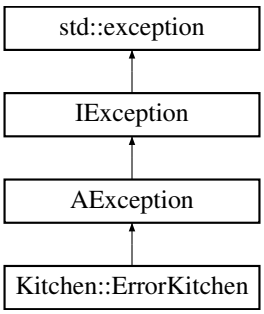
- **AException** (const std::string &type, const std::string &message)
- const char * what () const noexcept override
- std::string getType () const noexcept override
- std::string getMessage () const noexcept override
- std::string getFormattedMessage () const noexcept override

The documentation for this class was generated from the following files:

- src/cooks/Cooks.hpp
- src/cooks/ErrorCooks.cpp

5.10 Kitchen::ErrorKitchen Class Reference

Inheritance diagram for Kitchen::ErrorKitchen:



Public Member Functions

- **ErrorKitchen** (const std::string &message)

Public Member Functions inherited from AException

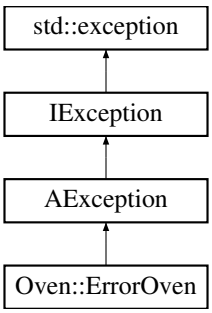
- **AException** (const std::string &type, const std::string &message)
- const char * what () const noexcept override
- std::string getType () const noexcept override
- std::string getMessage () const noexcept override
- std::string getFormattedMessage () const noexcept override

The documentation for this class was generated from the following files:

- src/kitchen/Kitchen.hpp
- src/kitchen/ErrorKitchen.cpp

5.11 Oven::ErrorOven Class Reference

Inheritance diagram for Oven::ErrorOven:



Public Member Functions

- **ErrorOven** (const std::string &message)

Public Member Functions inherited from AException

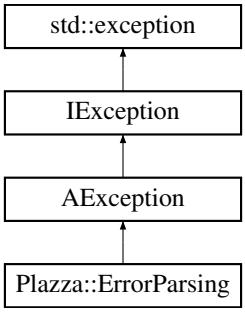
- **AException** (const std::string &type, const std::string &message)
- const char * what () const noexcept override
- std::string getType () const noexcept override
- std::string getMessage () const noexcept override
- std::string getFormattedMessage () const noexcept override

The documentation for this class was generated from the following files:

- src/kitchen/Oven.hpp
- src/kitchen/ErrorOven.cpp

5.12 Plazza::ErrorParsing Class Reference

Inheritance diagram for Plazza::ErrorParsing:



Public Member Functions

- **ErrorParsing** (const std::string &message)

Public Member Functions inherited from AException

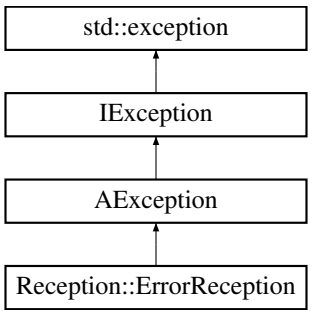
- **AException** (const std::string &type, const std::string &message)
- const char * what () const noexcept override
- std::string getType () const noexcept override
- std::string getMessage () const noexcept override
- std::string getFormattedMessage () const noexcept override

The documentation for this class was generated from the following files:

- src/Plazza.hpp
- src/ErrorParsing.cpp

5.13 Reception::ErrorReception Class Reference

Inheritance diagram for Reception::ErrorReception:



Public Member Functions

- **ErrorReception** (const std::string &message)

Public Member Functions inherited from AException

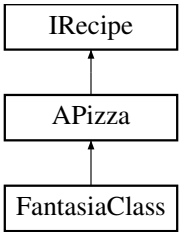
- **AException** (const std::string &type, const std::string &message)
- const char * what () const noexcept override
- std::string getType () const noexcept override
- std::string getMessage () const noexcept override
- std::string getFormattedMessage () const noexcept override

The documentation for this class was generated from the following files:

- src/reception/Reception.hpp
- src/reception/ErrorReception.cpp

5.14 FantasiaClass Class Reference

Inheritance diagram for FantasiaClass:



Public Member Functions

- **FantasiaClass** (int number)
- void cook (int cookTime) override
- std::shared_ptr< Ingridient > prepare (int number, std::shared_ptr< Ingridient > ingridient) override
- void serve () override

Public Member Functions inherited from APizza

- **APizza** (int number)
- int getNumber () const override
- void setNumber (int number) override

5.14.1 Member Function Documentation

5.14.1.1 cook()

```
void FantasiaClass::cook (
            int cookTime )  [override], [virtual]
```

Implements APizza.

5.14.1.2 prepare()

```
std::shared_ptr< Ingridient > FantasiaClass::prepare (
            int number,
            std::shared_ptr< Ingridient > ingridient )  [override], [virtual]
```

Implements APizza.

5.14.1.3 serve()

```
void FantasiaClass::serve ( )  [override], [virtual]
```

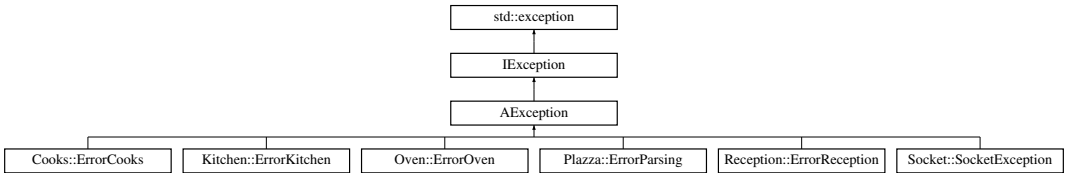
Implements APizza.

The documentation for this class was generated from the following files:

- src/recipes/pizza/Fantasia.hpp
- src/recipes/pizza/Fantasia.cpp

5.15 IException Class Reference

Inheritance diagram for IException:



Public Member Functions

- const char * **what** () const noexcept override=0
- virtual std::string **getType** () const noexcept=0
- virtual std::string **getMessage** () const noexcept=0
- virtual std::string **getFormattedMessage** () const noexcept=0

The documentation for this class was generated from the following file:

- common/IException.hpp

5.16 Ingridient Class Reference

Public Member Functions

- std::vector< ingStat > **fridgeStatus** ()
- int **getDough** () const
- int **getTomato** () const
- int **getCheese** () const
- int **getHam** () const
- int **getMushroom** () const
- int **getSteak** () const
- int **getEggplant** () const
- int **getGoatCheese** () const
- int **getChefLove** () const
- int **getEgg** () const
- int **getBacon** () const
- int **getBasil** () const
- int **getPepper** () const
- void **setDough** (int dough)
- void **setTomato** (int tomato)
- void **setCheese** (int cheese)
- void **setHam** (int ham)
- void **setMushroom** (int mushroom)
- void **setSteak** (int steak)
- void **setEggplant** (int eggplant)
- void **setGoatCheese** (int goatCheese)
- void **setChefLove** (int chefLove)
- void **setEgg** (int egg)
- void **setBacon** (int bacon)
- void **setBasil** (int basil)
- void **setPepper** (int pepper)
- std::string **packIngredients** () const

Static Public Member Functions

- static std::map< IngridientType, int > **unpackIngredients** (const std::string &packedData)

Private Attributes

- int **_dough**
- int **_tomato**
- int **_cheese**
- int **_ham**
- int **_mushroom**
- int **_steak**
- int **_eggplant**
- int **_goatCheese**
- int **_chefLove**
- int **_egg**
- int **_bacon**
- int **_basil**
- int **_pepper**
- std::vector< ingStat > **_ingridient**

The documentation for this class was generated from the following files:

- common/Ingridient.hpp
- common/Ingridient.cpp

5.17 ingStat Struct Reference

Public Attributes

- IngridientType **type**
- int **quantity**

The documentation for this struct was generated from the following file:

- common/Ingridient.hpp

5.18 IPCSocket Class Reference

Public Member Functions

- **IPCSocket** (std::string path)
- ssize_t **sendMessage** (const std::string &msg)
- std::string **recvMessage** ()

Private Attributes

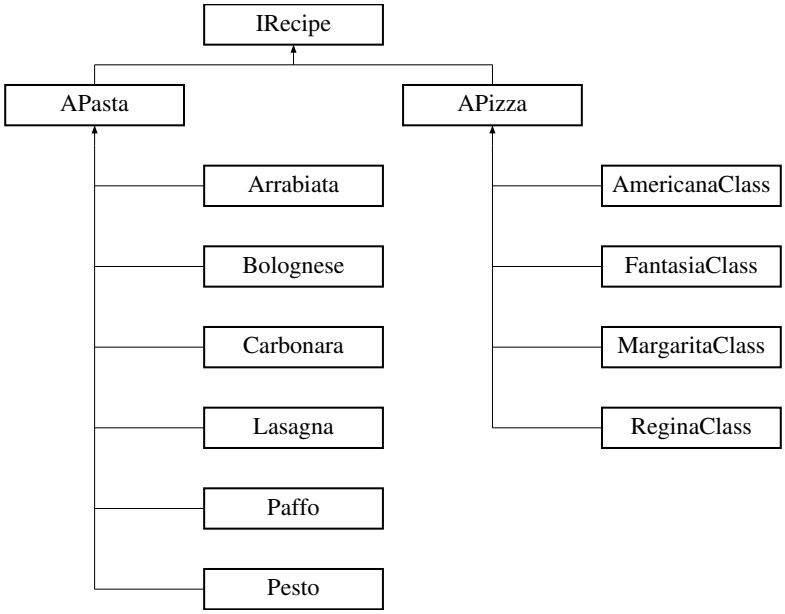
- int **_fd**
- std::string **_path**

The documentation for this class was generated from the following files:

- common/IPCSocket.hpp
- common/IPCSocket.cpp

5.19 IRecipe Class Reference

Inheritance diagram for IRecipe:



Public Member Functions

- virtual void **cook** (int cookTime)=0
- virtual std::shared_ptr< Ingridient > **prepare** (int number, std::shared_ptr< Ingridient > ingridient)=0
- virtual void **serve** ()=0
- virtual int **getNumber** () const =0
- virtual void **setNumber** (int number)=0

The documentation for this class was generated from the following file:

- common/IRecipe.hpp

5.20 Kitchen Class Reference

Classes

- class ErrorKitchen

Public Member Functions

- **Kitchen** (int id, int nbCooks, int cookTime, int restockTime, bool debug)
- void **startKitchenProcess** ()
- void **startKitchen** ()
- void **createCooks** ()
- void **restock** ()
- void **run** ()
- void **processOrder** (const std::string &orderData)
- bool **canAcceptOrder** (int numPizzas)
- void **stopKitchen** ()
- void **sendOrder** ()
- int **getID** () const
- int **getNbCooks** () const
- int **getCookTime** () const
- int **getRestockTime** () const
- int **getMaxCmd** () const
- std::shared_ptr< Ingridient > **getIngridient** () const
- std::vector< Cooks * > **getCooks** () const
- int **getCurrentOrders** () const

Private Attributes

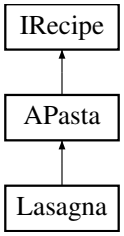
- int **_ID**
- int **_nbCooks**
- int **_cookTime**
- int **_restockTime**
- int **_maxCmd**
- int **_currentOrders**
- bool **_isDebug**
- bool **_isRunning**
- std::chrono::steady_clock::time_point **_lastActivity**
- std::mutex **_orderMutex**
- std::mutex **_ingMutex**
- std::condition_variable **_cookCV**
- Socket **_socket**
- std::shared_ptr< Ingridient > **_ingridient**
- std::vector< Cooks * > **_cooks**
- std::vector< std::thread > **_cookThreads**
- std::thread **_restockThread**
- std::queue< std::string > **_orderQueue**

The documentation for this class was generated from the following files:

- src/kitchen/Kitchen.hpp
- src/kitchen/Kitchen.cpp

5.21 Lasagna Class Reference

Inheritance diagram for Lasagna:



Public Member Functions

- **Lasagna** (int number)
- void cook (int cookTime) override
- std::shared_ptr< Ingridient > prepare (int number, std::shared_ptr< Ingridient > ingridient) override
- void serve () override

Public Member Functions inherited from APasta

- **APasta** (int number)
- int getNumber () const override
- void setNumber (int number) override

5.21.1 Member Function Documentation

5.21.1.1 cook()

```
void Lasagna::cook (
            int cookTime )  [override], [virtual]
```

Implements APasta.

5.21.1.2 prepare()

```
std::shared_ptr< Ingridient > Lasagna::prepare (
            int number,
            std::shared_ptr< Ingridient > ingridient )  [override], [virtual]
```

Implements APasta.

5.21.1.3 serve()

```
void Lasagna::serve ( )  [override], [virtual]
```

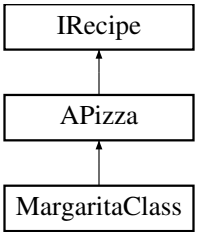
Implements APasta.

The documentation for this class was generated from the following files:

- src/recipes/pasta/Lasagna.hpp
- src/recipes/pasta/Lasagna.cpp

5.22 MargaritaClass Class Reference

Inheritance diagram for MargaritaClass:



Public Member Functions

- **MargaritaClass** (int number)
- void cook (int cookTime) override
- std::shared_ptr< Ingridient > prepare (int number, std::shared_ptr< Ingridient > ingridient) override
- void serve () override

Public Member Functions inherited from APizza

- **APizza** (int number)
- int getNumber () const override
- void setNumber (int number) override

5.22.1 Member Function Documentation

5.22.1.1 cook()

```
void MargaritaClass::cook (
            int cookTime )  [override], [virtual]
```

Implements APizza.

5.22.1.2 prepare()

```
std::shared_ptr< Ingridient > MargaritaClass::prepare (
            int number,
            std::shared_ptr< Ingridient > ingridient )  [override], [virtual]
```

Implements APizza.

5.22.1.3 serve()

```
void MargaritaClass::serve ( )  [override], [virtual]
```

Implements APizza.

The documentation for this class was generated from the following files:

- src/recipes/pizza/Margarita.hpp
- src/recipes/pizza/Margarita.cpp

5.23 Order Struct Reference

Static Public Member Functions

- static std::string **pack** (const Order &order)
- static Order **unpack** (const std::string &data)

Public Attributes

- PizzaType **type**
- Size **size**
- int **number**

The documentation for this struct was generated from the following file:

- common/Order.hpp

5.24 Oven Class Reference

Classes

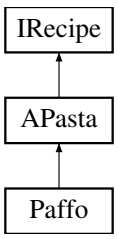
- class ErrorOven

The documentation for this class was generated from the following files:

- src/kitchen/Oven.hpp
- src/kitchen/Oven.cpp

5.25 Paffo Class Reference

Inheritance diagram for Paffo:



Public Member Functions

- **Paffo** (int number)
- void cook (int cookTime) override
- std::shared_ptr< Ingridient > prepare (int number, std::shared_ptr< Ingridient > ingridient) override
- void serve () override

Public Member Functions inherited from APasta

- **APasta** (int number)
- int getNumber () const override
- void setNumber (int number) override

5.25.1 Member Function Documentation

5.25.1.1 cook()

```
void Paffo::cook (
            int cookTime )  [override], [virtual]
```

Implements APasta.

5.25.1.2 prepare()

```
std::shared_ptr< Ingridient > Paffo::prepare (
            int number,
            std::shared_ptr< Ingridient > ingridient )  [override], [virtual]
```

Implements APasta.

5.25.1.3 serve()

```
void Paffo::serve ( )  [override], [virtual]
```

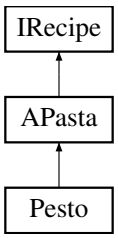
Implements APasta.

The documentation for this class was generated from the following files:

- src/recipes/pasta/Paffo.hpp
- src/recipes/pasta/Paffo.cpp

5.26 Pesto Class Reference

Inheritance diagram for Pesto:



Public Member Functions

- **Pesto** (int number)
- void cook (int cookTime) override
- std::shared_ptr< Ingridient > prepare (int number, std::shared_ptr< Ingridient > ingridient) override
- void serve () override

Public Member Functions inherited from APasta

- **APasta** (int number)
- int getNumber () const override
- void setNumber (int number) override

5.26.1 Member Function Documentation

5.26.1.1 cook()

```
void Pesto::cook (
            int cookTime )  [override], [virtual]
```

Implements APasta.

5.26.1.2 prepare()

```
std::shared_ptr< Ingridient > Pesto::prepare (
            int number,
            std::shared_ptr< Ingridient > ingridient )  [override], [virtual]
```

Implements APasta.

5.26.1.3 serve()

```
void Pesto::serve ( )  [override], [virtual]
```

Implements APasta.

The documentation for this class was generated from the following files:

- src/recipes/pasta/Pesto.hpp
- src/recipes/pasta/Pesto.cpp

5.27 Plazza Class Reference

Classes

- class ErrorParsing

Public Member Functions

- void **parseCmd** (char ∗∗av, int ac)
- void **orderingLoop** ()

Private Attributes

- int **_nbCooks**
- int **_timerCooker**
- int **_timerRestock**
- bool **_debug**
- Reception **_reception**

The documentation for this class was generated from the following files:

- src/Plazza.hpp
- src/Plazza.cpp

5.28 Reception Class Reference

Classes

- class ErrorReception

Public Member Functions

- void **setValues** (int nbCooks, int cookTime, int restockTime, bool debug)
- void **createKitchen** (int id, int nbCooks, int cookTime, int restockTime)
- void **destroyKitchen** (int id)
- void **orderingLoop** ()
- void **interMessaege** (std::shared_ptr< Socket > socket, int id)
- std::vector< std::string > **checkCommand** (const char ∗command)
- void **processOrders** (const std::vector< std::string > &orders)
- bool **sendOrderToKitchen** (const std::string &orderData)
- void **monitorKitchens** ()
- void **updateKitchenStat** (std::map< IngridientType, int > ingredients, std::shared_ptr< Kitchen > kitchens)
- int **getNbKitchens** () const
- std::vector< std::shared_ptr< Kitchen > > **getKitchens** () const
- std::shared_ptr< Kitchen > **getKitchen** (int id) const

Private Attributes

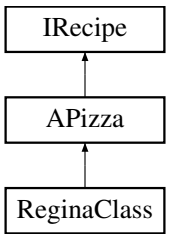
- int **_nbCooks**
- int **_cookTime**
- int **_restockTime**
- int **_nbKitchens**
- bool **_isDebug**
- bool **_isRunning**
- std::vector< std::shared_ptr< Kitchen > > **_kitchens**
- std::map< int, std::shared_ptr< Socket > > **_kitchenSockets**
- std::mutex **_kitchensMutex**
- std::thread **_monitorThread**

The documentation for this class was generated from the following files:

- src/reception/Reception.hpp
- src/reception/CommandParser.cpp
- src/reception/ReceiveMessageKitchen.cpp
- src/reception/Reception.cpp

5.29 ReginaClass Class Reference

Inheritance diagram for ReginaClass:



Public Member Functions

- **ReginaClass** (int number)
- void cook (int cookTime) override
- std::shared_ptr< Ingridient > prepare (int number, std::shared_ptr< Ingridient > ingridient) override
- void serve () override

Public Member Functions inherited from APizza

- **APizza** (int number)
- int getNumber () const override
- void setNumber (int number) override

5.29.1 Member Function Documentation

5.29.1.1 cook()

```
void ReginaClass::cook (
            int cookTime )  [override], [virtual]
```

Implements APizza.

5.29.1.2 prepare()

```
std::shared_ptr< Ingridient > ReginaClass::prepare (
            int number,
            std::shared_ptr< Ingridient > ingridient )  [override], [virtual]
```

Implements APizza.

5.29.1.3 serve()

```
void ReginaClass::serve ( )  [override], [virtual]
```

Implements APizza.

The documentation for this class was generated from the following files:

- src/recipes/pizza/Regina.hpp
- src/recipes/pizza/Regina.cpp

5.30 Socket Class Reference

Classes

- class SocketException

Public Member Functions

- void **createServer** (const std::string &sockPath)
- void **acceptClient** ()
- void **closeServer** ()
- void **connectToServer** (const std::string &sockPath)
- void **closeClient** ()
- ssize_t **send** (const std::string &message)
- std::string **receive** (size_t size=1024)
- bool **isConnected** () const
- Socket & **operator**<< (const std::string &message)
- Socket & **operator**>> (std::string &message)

Private Attributes

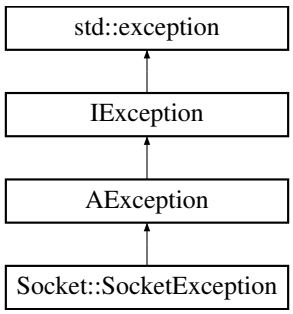
- int **_serverFd**
- int **_clientFd**
- struct sockaddr_un **_addr**
- std::string **_sockPath**
- bool **_isServer**
- bool **_isConnected**

The documentation for this class was generated from the following files:

- common/Socket.hpp
- common/Socket.cpp

5.31 Socket::SocketException Class Reference

Inheritance diagram for Socket::SocketException:



Public Member Functions

- **SocketException** (const std::string &message)

Public Member Functions inherited from AException

- **AException** (const std::string &type, const std::string &message)
- const char * what () const noexcept override
- std::string getType () const noexcept override
- std::string getMessage () const noexcept override
- std::string getFormattedMessage () const noexcept override

The documentation for this class was generated from the following files:

- common/Socket.hpp
- common/ErrorSocket.cpp

5.32 Utils Class Reference

Public Member Functions

- void **helper** ()

The documentation for this class was generated from the following files:

- src/utils/Utils.hpp
- src/utils/Utils.cpp

Chapter 6

File Documentation

6.1 AException.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** AExeption
00006 */
00007
00008 #ifndef AEXEPTION_HPP_
00009     #define AEXEPTION_HPP_
00010
00011 #include "IException.hpp"
00012 #include <string>
00013
00014 class AException : public IException {
00015     public:
00016         AException(const std::string& type, const std::string& message)
00017         : _message(message), _type(type) {}
00018         virtual ~AException() noexcept = default;
00019
00020         const char* what() const noexcept override {
00021             return getFormattedMessage().c_str();
00022         }
00023
00024         std::string getType() const noexcept override {
00025             return _type;
00026         }
00027
00028         std::string getMessage() const noexcept override {
00029             return _message;
00030         }
00031
00032         std::string getFormattedMessage() const noexcept override {
00033             return "\033[1;31m[" + _type + "]\033[0m " + _message;
00034         }
00035
00036     private:
00037         std::string _message;
00038         std::string _type;
00039 };
00040
00041 #endif /* !AEXEPTION_HPP_ */
```

6.2 APasta.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** APasta
00006 */
00007
00008 #ifndef APASTA_HPP_
00009 #define APASTA_HPP_
00010
```

```
00011 #include "IRecipe.hpp"
00012
00013 enum PastaType
00014 {
00015     Carbonara = 1,
00016     Pesto = 2,
00017     Bolognese = 3,
00018     Arrabiata = 4,
00019     Paffo = 6,
00020     Lasagna = 10
00021 };
00022
00023
00024 class APasta : public IRecipe {
00025     public:
00026         APasta(int number);
00027         virtual ~APasta() override = default;
00028         virtual void cook(int cookTime) override = 0;
00029         virtual std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient>
      ingridient) override = 0;
00030         virtual void serve() override = 0;
00031
00032         /* Getter */
00033         int getNumber() const override;
00034
00035         /* Setter */
00036         void setNumber(int number) override;
00037
00038     private:
00039         int _size;
00040         int _number;
00041 };
00042
00043 #endif /* !APASTA_HPP_ */
```

6.3 APizza.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** APizza
00006 */
00007
00008 #ifndef APIZZA_HPP_
00009 #define APIZZA_HPP_
00010
00011 #include "IRecipe.hpp"
00012
00013 enum PizzaType
00014 {
00015     Nothing = 0,
00016     Regina = 1,
00017     Margarita = 2,
00018     Americana = 4,
00019     Fantasia = 8
00020 };
00021
00022
00023 class APizza : public IRecipe {
00024     public:
00025         APizza(int number);
00026         virtual ~APizza() override = default;
00027         virtual void cook(int cookTime) override = 0;
00028         virtual std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient>
      ingridient) override = 0;
00029         virtual void serve() override = 0;
00030
00031         /* Getter */
00032         int getNumber() const override;
00033
00034         /* Setter */
00035         void setNumber(int number) override;
00036     private:
00037         int _number;
00038
00039 };
00040
00041 #endif /* !APIZZA_HPP_ */
```

6.4 IException.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** IExeption
00006 */
00007
00008 #include <exception>
00009 #include <string>
00010
00011 #ifndef IEXEPTION_HPP_
00012     #define IEXEPTION_HPP_
00013
00014 class IException : public std::exception {
00015     public:
00016         virtual ~IException() noexcept = default;
00017         const char* what() const noexcept override = 0;
00018         virtual std::string getType() const noexcept = 0;
00019         virtual std::string getMessage() const noexcept = 0;
00020         virtual std::string getFormattedMessage() const noexcept = 0;
00021 };
00022
00023 #endif /* !IEXEPTION_HPP_ */
```

6.5 Ingridient.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** Ingridient
00006 */
00007
00008 #include <vector>
00009 #include <string>
00010 #include <sstream>
00011 #include <map>
00012
00013 #ifndef INGRIDIENT_HPP_
00014 #define INGRIDIENT_HPP_
00015
00016 enum IngridientType
00017 {
00018     DOUGH = 0,
00019     TOMATO = 1,
00020     CHEESE = 2,
00021     HAM = 3,
00022     MUSHROOM = 4,
00023     STEAK = 5,
00024     EGGPLANT = 6,
00025     GOAT_CHEESE = 7,
00026     CHEF_LOVE = 8,
00027     EGG = 9,
00028     BACON = 10,
00029     BASIL = 11,
00030     PEPPER = 12
00031 };
00032
00033 struct ingStat {
00034     IngridientType type;
00035     int quantity;
00036 };
00037
00038 class Ingridient {
00039     public:
00040         Ingridient();
00041         ~Ingridient() = default;
00042         std::vector<ingStat> fridgeStatus();
00043
00044         /* Getter */
00045         int getDough() const;
00046         int getTomato() const;
00047         int getCheese() const;
00048         int getHam() const;
00049         int getMushroom() const;
00050         int getSteak() const;
00051         int getEggplant() const;
00052         int getGoatCheese() const;
00053         int getChefLove() const;
00054         int getEgg() const;
00055         int getBacon() const;
```

```
00056         int getBasil() const;
00057         int getPepper() const;
00058
00059         /* Setter */
00060         void setDough(int dough);
00061         void setTomato(int tomato);
00062         void setCheese(int cheese);
00063         void setHam(int ham);
00064         void setMushroom(int mushroom);
00065         void setSteak(int steak);
00066         void setEggplant(int eggplant);
00067         void setGoatCheese(int goatCheese);
00068         void setChefLove(int chefLove);
00069         void setEgg(int egg);
00070         void setBacon(int bacon);
00071         void setBasil(int basil);
00072         void setPepper(int pepper);
00073
00074         /* Packing/Unpacking methods */
00075         std::string packIngredients() const;
00076         static std::map<IngridientType, int> unpackIngredients(const std::string& packedData);
00077
00078     private:
00079         int _dough;
00080         int _tomato;
00081         int _cheese;
00082         int _ham;
00083         int _mushroom;
00084         int _steak;
00085         int _eggplant;
00086         int _goatCheese;
00087         int _chefLove;
00088         int _egg;
00089         int _bacon;
00090         int _basil;
00091         int _pepper;
00092         std::vector<ingStat> _ingridient;
00093 };
00094
00095 #endif /* !INGRIDIENT_HPP_ */
```

6.6 IPCSocket.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** IPCSocket
00006 */
00007
00008 #ifndef IPCSOCKET_HPP_
00009 #define IPCSOCKET_HPP_
00010
00011 #include <sys/socket.h>
00012 #include <sys/un.h>
00013 #include <unistd.h>
00014 #include <iostream>
00015
00016 #include "IRecipe.hpp"
00017
00018 class IPCSocket {
00019     public:
00020         IPCSocket(std::string path);
00021         ~IPCSocket() = default;
00022         ssize_t sendMessage(const std::string& msg);
00023         std::string recvMessage();
00024
00025         /* Overload */
00026         // IPCSocket &operator«(const IPCSocket &socket, const IReceipy &recipy);
00027         // IPCSocket &operator»(const IPCSocket &socket,const IReceipy &recipy);
00028     protected:
00029     private:
00030         int _fd;
00031         std::string _path;
00032 };
00033
00034
00035 #endif /* !IPCSOCKET_HPP_ */
```

6.7 IRecipe.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** IReceipy
00006 */
00007
00008
00009 #include <memory>
00010 #include "Ingridient.hpp"
00011
00012 #ifndef IRECIPE_HPP_
00013 #define IRECIPE_HPP_
00014
00015 enum Size
00016 {
00017     Zero = 0,
00018     S = 1,
00019     M = 2,
00020     L = 4,
00021     XL = 8,
00022     XXL = 16
00023 };
00024
00025
00026 class IRecipe {
00027     public:
00028
00029         virtual ~IRecipe() = default;
00030         virtual void cook(int cookTime) = 0;
00031         virtual std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient>
      ingridient) = 0;
00032         virtual void serve() = 0;
00033
00034         /* Getter */
00035         virtual int getNumber() const = 0;
00036         /* Setter */
00037         virtual void setNumber(int number) = 0;
00038
00039 };
00040
00041 #endif /* !IRECEIPY_HPP_ */
```

6.8 Order.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** Order
00006 */
00007
00008 #include <iostream>
00009 #include <sstream>
00010 #include <regex>
00011 #include <string>
00012
00013 #include "APizza.hpp"
00014
00015 struct Order {
00016     PizzaType type;
00017     Size size;
00018     int number;
00019
00020     static std::string pack(const Order &order) {
00021         std::ostringstream oss;
00022         oss « "0x01|" « static_cast<int>(order.type) « "|"
00023             « static_cast<int>(order.size) « "|"
00024             « order.number « ";";
00025         return oss.str();
00026     }
00027
00028     static Order unpack(const std::string &data) {
00029         Order order = {Nothing, Zero, 0};
00030         std::regex pattern("0x01\\|(\\d+)\\|(\\d+)\\|(\\d+);");
00031         std::smatch match;
00032         if (std::regex_match(data, match, pattern)) {
00033             order.type = static_cast<PizzaType>(std::stoi(match[1].str()));
00034             order.size = static_cast<Size>(std::stoi(match[2].str()));
00035             order.number = std::stoi(match[3].str());
00036         }
```

```
00037         return order;
00038     }
00039 };
```

6.9 Socket.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** Socket
00006 */
00007
00008 #ifndef SOCKET_HPP_
00009 #define SOCKET_HPP_
00010
00011 #include <sys/socket.h>
00012 #include <sys/un.h>
00013 #include <string>
00014 #include <unistd.h>
00015 #include <memory>
00016 #include "AException.hpp"
00017
00018 class Socket {
00019
00020     public:
00021         class SocketException : public AException {
00022             public:
00023                 SocketException(const std::string &message);
00024         };
00025
00026         Socket();
00027         ~Socket();
00028
00029         // Server operations
00030         void createServer(const std::string &sockPath);
00031         void acceptClient();
00032         void closeServer();
00033
00034         // Client operations
00035         void connectToServer(const std::string &sockPath);
00036         void closeClient();
00037
00038         // Common operations
00039         ssize_t send(const std::string &message);
00040         std::string receive(size_t size = 1024);
00041         bool isConnected() const;
00042
00043         // Operators
00044         Socket& operator«(const std::string &message);
00045         Socket& operator»(std::string &message);
00046
00047     private:
00048         int _serverFd;
00049         int _clientFd;
00050         struct sockaddr_un _addr;
00051         std::string _sockPath;
00052         bool _isServer;
00053         bool _isConnected;
00054 };
00055
00056 #endif /* !SOCKET_HPP_ */
```

6.10 Cooks.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** Cooks
00006 */
00007
00008 #include <iostream>
00009 #include <memory>
00010 #include <mutex>
00011 #include <vector>
00012
00013 #include "../../common/AException.hpp"
00014 #include "../../common/Ingridient.hpp"
```

```
00015
00016 #ifndef COOKS_HPP_
00017     #define COOKS_HPP_
00018
00019 class Cooks {
00020
00021     class ErrorCooks : public AException {
00022         public:
00023             ErrorCooks(const std::string &message);
00024     };
00025
00026     public:
00027         Cooks(std::shared_ptr<Ingridient> ingridient, int id,
00028             int cookTime, int restockTime);
00029         ~Cooks() = default;
00030
00031         /* Method */
00032         std::shared_ptr<Ingridient> startOrder(std::shared_ptr<Ingridient> ingridient,
      std::vector<std::string> order);
00033         void restock();
00034         bool hasEnoughIngredients(const std::string &orderData, std::shared_ptr<Ingridient>
      ingridient);
00035
00036         /* Getter */
00037         int getID() const;
00038         bool isBusy() const;
00039         bool isRestocking() const;
00040
00041     private:
00042         int _ID;
00043         int _cookTime;
00044         int _restockTime;
00045         std::shared_ptr<Ingridient> _ingridient;
00046         std::mutex _statusMutex;
00047         bool _isBusy;
00048         bool _isRestocking;
00049 };
00050
00051 #endif /* !COOKS_HPP_ */
```

6.11 Kitchen.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** Kitchen
00006 */
00007
00008 #ifndef KITCHEN_HPP_
00009 #define KITCHEN_HPP_
00010
00011 #include <iostream>
00012 #include <string>
00013 #include <vector>
00014 #include <queue>
00015 #include <memory>
00016 #include <mutex>
00017 #include <condition_variable>
00018 #include <chrono>
00019 #include <thread>
00020
00021 #include "../../common/Ingridient.hpp"
00022 #include "../../common/Socket.hpp"
00023 #include "../cooks/Cooks.hpp"
00024
00025 class Kitchen {
00026
00027     class ErrorKitchen : public AException {
00028         public:
00029             ErrorKitchen(const std::string &message);
00030     };
00031
00032     public:
00033         Kitchen(int id, int nbCooks, int cookTime, int restockTime, bool debug);
00034         ~Kitchen();
00035
00036         void startKitchenProcess();
00037         void startKitchen();
00038         void createCooks();
00039         void restock();
00040         void run();
00041         void processOrder(const std::string &orderData);
```

```
00042         bool canAcceptOrder(int numPizzas);
00043         void stopKitchen();
00044         void sendOrder();
00045
00046         /* Getter */
00047         int getID() const;
00048         int getNbCooks() const;
00049         int getCookTime() const;
00050         int getRestockTime() const;
00051         int getMaxCmd() const;
00052         std::shared_ptr<Ingridient> getIngridient() const;
00053         std::vector<Cooks*> getCooks() const;
00054         int getCurrentOrders() const;
00055
00056     private:
00057         int _ID;
00058         int _nbCooks;
00059         int _cookTime;
00060         int _restockTime;
00061         int _maxCmd;
00062         int _currentOrders;
00063         bool _isDebug;
00064
00065         bool _isRunning;
00066         std::chrono::steady_clock::time_point _lastActivity;
00067         std::mutex _orderMutex;
00068         std::mutex _ingMutex;
00069         std::condition_variable _cookCV;
00070         Socket _socket;
00071
00072         std::shared_ptr<Ingridient> _ingridient;
00073         std::vector<Cooks*> _cooks;
00074         std::vector<std::thread> _cookThreads;
00075         std::thread _restockThread;
00076         std::queue<std::string> _orderQueue;
00077 };
00078
00079 std::ostream& operator«(std::ostream& os, const Kitchen& kitchen);
00080
00081 #endif /* !KITCHEN_HPP_ */
```

6.12 Oven.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** Oven
00006 */
00007
00008 #include <iostream>
00009 #include "../../common/AException.hpp"
00010
00011 #ifndef OVEN_HPP_
00012     #define OVEN_HPP_
00013
00014 class Oven {
00015
00016     class ErrorOven : public AException {
00017         public:
00018             ErrorOven(const std::string &message);
00019     };
00020
00021     public:
00022         Oven();
00023         ~Oven();
00024
00025     protected:
00026     private:
00027 };
00028
00029 #endif /* !OVEN_HPP_ */
```

6.13 Plazza.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
```

```
00005 ** Plazza
00006 */
00007
00008 #ifndef PLAZZA_HPP_
00009 #define PLAZZA_HPP_
00010
00011 #include "reception/Reception.hpp"
00012 #include "../common/AException.hpp"
00013
00014 class Plazza {
00015
00016     class ErrorParsing : public AException {
00017         public:
00018             ErrorParsing(const std::string &message);
00019     };
00020
00021     public:
00022         Plazza();
00023         ~Plazza();
00024
00025         void parseCmd(char **av, int ac);
00026         void orderingLoop();
00027
00028     private:
00029         int _nbCooks;
00030         int _timerCooker;
00031         int _timerRestock;
00032         bool _debug;
00033         Reception _reception;
00034 };
00035
00036 #endif /* !PLAZZA_HPP_ */
```

6.14 Reception.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane
00004 ** File description:
00005 ** Reception
00006 */
00007
00008 #include <vector>
00009 #include <sstream>
00010 #include <regex>
00011 #include <map>
00012 #include <thread>
00013 #include <mutex>
00014
00015 #include "../../common/AException.hpp"
00016 #include "../../common/APizza.hpp"
00017 #include "../../common/Socket.hpp"
00018 #include "../kitchen/Kitchen.hpp"
00019
00020
00021 #ifndef RECEPTION_HPP_
00022     #define RECEPTION_HPP_
00023
00024
00025 class Reception {
00026
00027     class ErrorReception : public AException {
00028         public:
00029             ErrorReception(const std::string &message);
00030     };
00031
00032     public:
00033         Reception();
00034         ~Reception();
00035
00036         /* Methods */
00037         void setValues(int nbCooks, int cookTime, int restockTime, bool debug);
00038         void createKitchen(int id, int nbCooks, int cookTime, int restockTime);
00039         void destroyKitchen(int id);
00040         void orderingLoop();
00041         void interMessaege(std::shared_ptr<Socket> socket, int id);
00042         std::vector<std::string> checkCommand(const char *command);
00043         void processOrders(const std::vector<std::string> &orders);
00044         bool sendOrderToKitchen(const std::string &orderData);
00045         void monitorKitchens();
00046         void updateKitchenStat(std::map<IngridientType, int> ingredients,
00047             std::shared_ptr<Kitchen> kitchens);
00048
```

```
00049         /* Getter */
00050         int getNbKitchens() const;
00051         std::vector<std::shared_ptr<Kitchen» getKitchens() const;
00052         std::shared_ptr<Kitchen> getKitchen(int id) const;
00053
00054     private:
00055         int _nbCooks;
00056         int _cookTime;
00057         int _restockTime;
00058         int _nbKitchens;
00059         bool _isDebug;
00060         bool _isRunning;
00061         std::vector<std::shared_ptr<Kitchen» _kitchens;
00062         std::map<int, std::shared_ptr<Socket» _kitchenSockets;
00063         std::mutex _kitchensMutex;
00064         std::thread _monitorThread;
00065 };
00066
00067 /* Pizza Order overloader */
00068
00069 #endif /* !RECEPTION_HPP_ */
```

6.15 Arrabiata.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** The Plazza
00004 ** File description:
00005 ** Arrabiata
00006 */
00007
00008 #include "../../../common/APasta.hpp"
00009
00010 #ifndef ARRABIATA_HPP_
00011     #define ARRABIATA_HPP_
00012
00013 class Arrabiata : public APasta {
00014     public:
00015         Arrabiata(int number);
00016         ~Arrabiata() override;
00017
00018         /* Method */
00019         void cook(int cookTime) override;
00020         std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient> ingridient)
      override;
00021         void serve() override;
00022
00023     protected:
00024     private:
00025 };
00026
00027 #endif /* !ARRABIATA_HPP_ */
```

6.16 Bolognese.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** The Plazza
00004 ** File description:
00005 ** Boloss
00006 */
00007
00008 #include "../../../common/APasta.hpp"
00009
00010 #ifndef BOLOGNESE_HPP_
00011     #define BOLOGNESE_HPP_
00012
00013 class Bolognese : public APasta {
00014     public:
00015         Bolognese(int number);
00016         ~Bolognese() override;
00017
00018         /* Method */
00019         void cook(int cookTime);
00020         std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient> ingridient);
00021         void serve();
00022
00023     private:
00024 };
00025
00026 #endif /* !BOLOGNESE_HPP_ */
```

6.17 Carbonara.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** Carbonara
00006 */
00007
00008 #include "../../../common/APasta.hpp"
00009
00010 #ifndef CARBONARA_HPP_
00011     #define CARBONARA_HPP_
00012
00013 class Carbonara : public APasta {
00014     public:
00015         Carbonara(int number);
00016         ~Carbonara() override;
00017
00018         /* Method */
00019         void cook(int cookTime) override;
00020         std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient> ingridient)
      override;
00021         void serve() override;
00022
00023     private:
00024 };
00025
00026 #endif /* !CARBONAR_HPP_ */
```

6.18 Lasagna.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** The Plazza
00004 ** File description:
00005 ** Lasagna
00006 */
00007
00008 #include "../../../common/APasta.hpp"
00009
00010 #ifndef LASAGNA_HPP_
00011     #define LASAGNA_HPP_
00012
00013 class Lasagna : public APasta {
00014     public:
00015         Lasagna(int number);
00016         ~Lasagna() override;
00017
00018         /* Method */
00019         void cook(int cookTime) override;
00020         std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient> ingridient)
      override;
00021         void serve() override;
00022
00023     private:
00024 };
00025
00026 #endif /* !LASAGNA_HPP_ */
```

6.19 Paffo.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** The Plazza
00004 ** File description:
00005 ** Paffo
00006 */
00007
00008 #include "../../../common/APasta.hpp"
00009
00010 #ifndef PAFFO_HPP_
00011     #define PAFFO_HPP_
00012
00013 class Paffo : public APasta {
00014     public:
00015         Paffo(int number);
00016         ~Paffo();
00017
```

```
00018         /* Method */
00019         void cook(int cookTime) override;
00020         std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient> ingridient)
      override;
00021         void serve() override;
00022
00023     private:
00024 };
00025
00026 #endif /* !PAFFO_HPP_ */
```

6.20 Pesto.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** Friteuse
00004 ** File description:
00005 ** Pesto
00006 */
00007
00008 #include "../../../common/APasta.hpp"
00009
00010 #ifndef PESTO_HPP_
00011     #define PESTO_HPP_
00012
00013 class Pesto : public APasta {
00014     public:
00015         Pesto(int number);
00016         ~Pesto() override;
00017
00018         /* Method */
00019         void cook(int cookTime) override;
00020         std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient> ingridient)
      override;
00021         void serve() override;
00022
00023     private:
00024 };
00025
00026 #endif /* !PESTO_HPP_ */
```

6.21 Americana.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** AmericanaClass
00006 */
00007
00008
00009 #include "../../../common/APizza.hpp"
00010
00011 #ifndef AMERICANA_HPP_
00012 #define AMERICANA_HPP_
00013
00014 class AmericanaClass : public APizza {
00015     public:
00016         AmericanaClass(int number);
00017         ~AmericanaClass() override;
00018
00019         /* Method */
00020         void cook(int cookTime) override;
00021         std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient> ingridient)
      override;
00022         void serve() override;
00023
00024     protected:
00025     private:
00026 };
00027
00028 #endif /* !AMERICANA_HPP_ */
```

6.22 Fantasia.hpp

```
00001 /*
```

```
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** FantasiaClass
00006 */
00007
00008
00009 #include "../../../common/APizza.hpp"
00010
00011 #ifndef FANTASIA_HPP_
00012 #define FANTASIA_HPP_
00013
00014 class FantasiaClass : public APizza {
00015     public:
00016         FantasiaClass(int number);
00017         ~FantasiaClass() override;
00018
00019         void cook(int cookTime) override;
00020         std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient> ingridient)
      override;
00021         void serve() override;
00022
00023     protected:
00024     private:
00025 };
00026
00027 #endif /* !FANTASIA_HPP_ */
```

6.23 Margarita.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** MargaritaCLASS
00006 */
00007
00008 #include "../../../common/APizza.hpp"
00009
00010 #ifndef MARGARITACLASS_HPP_
00011 #define MARGARITACLASS_HPP_
00012
00013 class MargaritaClass : public APizza {
00014     public:
00015         MargaritaClass(int number);
00016         ~MargaritaClass() override;
00017
00018         void cook(int cookTime) override;
00019         std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient> ingridient)
      override;
00020         void serve() override;
00021
00022
00023     protected:
00024     private:
00025 };
00026
00027 #endif /* !MARGARITACLASS_HPP_ */
```

6.24 Regina.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** ReginaClass
00006 */
00007
00008
00009 #include "../../../common/APizza.hpp"
00010
00011 #ifndef REGINACLASS_HPP_
00012 #define REGINACLASS_HPP_
00013
00014 class ReginaClass : public APizza {
00015     public:
00016         ReginaClass(int number);
00017         ~ReginaClass() override;
00018
```

```
00019         void cook(int cookTime) override;
00020         std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient> ingridient)
    override;
00021         void serve() override;
00022     protected:
00023     private:
00024 };
00025
00026 #endif /* !REGINACLASS_HPP_ */
```

6.25 Utils.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** Utils
00006 */
00007
00008 #ifndef UTILS_HPP_
00009 #define UTILS_HPP_
00010
00011 class Utils {
00012     public:
00013         Utils() = default;
00014         ~Utils() = default;
00015
00016         void helper();
00017
00018     protected:
00019     private:
00020 };
00021
00022 #endif /* !UTILS_HPP_ */
```