

Arcade architecture

Generated by Doxygen 1.10.0

Chapter 1

Project Detail

Plazza Project

The point of the plazza project is to experiment with multi-process / multi-threading execution. A reception will be defined which works like a shell in which we can order pizzas, to cook those pizzas a kitchen will need to be created, each kitchen will have a pre-defined number of cooks that can have two pizzas in their queues.

Libraries

Only authorized:
Standard C++ library

1.1 Bonus :

- Interface implementation of recipe(Pasta)
- Dynamic reload of parsing / recipe
- Size impact the cooking time
- Menu command to see what to order
- extensive debug printing

1.2 Compilation :

- make / make re
- make clean / make fclean
- make coding

1.3 Coding Style :

The Cpp code needs to abide to a specified coding style, to check if the code is compliant with the norm execute the make coding command or the ./styleChecker.sh. To understand the errors and how to fix them please refer to the coding-cpp.txt.

1.4 Documentation :

1.4.1 Docusorus :

To start the docusarus documentation : `cd documentation/my-website npx docusaurus start`

1.4.2 Doxygen :

The basic documentation fo the project is generated using the doxygen, to run the doxygen executable, please make sure you installed the pdf-latex librairie. To generate the PDF : `./generateDoc.sh`

1.5 Commit norm :

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

Gitmoji = The emoji appropriate 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

...

1.5.1 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!

1.6 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 fichier 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:

Cooks	??
CookStatusData	??
std::exception	
IException	??
AException	??
Cooks::ErrorCooks	??
Cooks::ErrorCooks	??
Kitchen::ErrorKitchen	??
Kitchen::ErrorKitchen	??
Plazza::ErrorParsing	??
Plazza::ErrorParsing	??
Reception::ErrorReception	??
Reception::ErrorReception	??
Socket::SocketException	??
Socket::SocketException	??
AException	??
IException	??
ILoader	??
DLLoader< T >	??
DLLoader< T >	??
IMesagges	??
AStatus	??
DoneStatus	??
DoneStatus	??
RefillStatus	??
RefillStatus	??
AStatus	??
CookStatus	??
CookStatus	??
Inactivity	??
Inactivity	??
Order	??
Order	??
Queue	??

Queue	??
Ingridient	??
ingStat	??
IProcess	??
ForkProcess	??
ForkProcess	??
MockProcess	??
IRecipe	??
APasta	??
ArrabiataClass	??
BologneseClass	??
CarbonaraClass	??
LasagnaClass	??
PaffoClass	??
PestoClass	??
TestPasta	??
APizza	??
AmericanaClass	??
AmericanaClass	??
FantasiaClass	??
FantasiaClass	??
MargaritaClass	??
MargaritaClass	??
ReginaClass	??
ReginaClass	??
TestPizza	??
APizza	??
Kitchen	??
MockSocket	??
Plazza	??
Reception	??
std::runtime_error	
ForkProcess::ProcessException	??
ForkProcess::ProcessException	??
Socket	??
Utils	??

Chapter 3

Class Index

3.1 Class List

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

AException	??
AmericanaClass	??
APasta	??
APizza	??
ArrabiataClass	??
AStatus	??
BologneseClass	??
CarbonaraClass	??
Cooks	??
CookStatus	??
CookStatusData	??
DLLoader< T >	??
DoneStatus	??
Cooks::ErrorCooks	??
Kitchen::ErrorKitchen	??
Plazza::ErrorParsing	??
Reception::ErrorReception	??
FantasiaClass	??
ForkProcess	??
IException	??
ILoader	??
IMesagges	??
Inactivity	??
Ingridient	??
ingStat	??
IProcess	??
IRecipe	??
Kitchen	??
LasagnaClass	??
MargaritaClass	??
MockProcess	??
MockSocket	??
Order	??
PaffoClass	??
PestoClass	??

Plazza	??
ForkProcess::ProcessException	??
Queue	??
Reception	??
RefillStatus	??
ReginaClass	??
Socket	??
Socket::SocketException	??
TestPasta	??
TestPizza	??
Utils	??

Chapter 4

File Index

4.1 File List

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

bonus/common/AException.hpp	??
bonus/common/APasta.hpp	??
bonus/common/APizza.hpp	??
bonus/common/IException.hpp	??
bonus/common/Ingridient.hpp	??
bonus/common/IRecipe.hpp	??
bonus/common/Socket.hpp	??
bonus/common/messages/AStatus.hpp	??
bonus/common/messages/CookStatus.hpp	??
bonus/common/messages/DoneStatus.hpp	??
bonus/common/messages/IMesaggess.hpp	??
bonus/common/messages/Inactivity.hpp	??
bonus/common/messages/Order.hpp	??
bonus/common/messages/Queue.hpp	??
bonus/common/messages/RefillStatus.hpp	??
bonus/common/processes/ForkProcess.hpp	??
bonus/common/processes/IProcess.hpp	??
bonus/lib/DLoader.hpp	??
bonus/lib/Loader.hpp	??
bonus/src/Plazza.hpp	??
bonus/src/cooks/Cooks.hpp	??
bonus/src/kitchen/Kitchen.hpp	??
bonus/src/reception/Reception.hpp	??
bonus/src/recipes/pasta/Arrabiata.hpp	??
bonus/src/recipes/pasta/Bolognese.hpp	??
bonus/src/recipes/pasta/Carbonara.hpp	??
bonus/src/recipes/pasta/Lasagna.hpp	??
bonus/src/recipes/pasta/Paffo.hpp	??
bonus/src/recipes/pasta/Pesto.hpp	??
bonus/src/recipes/pizza/Americana.hpp	??
bonus/src/recipes/pizza/Fantasia.hpp	??
bonus/src/recipes/pizza/Margarita.hpp	??
bonus/src/recipes/pizza/Regina.hpp	??
bonus/src/utills/Utills.hpp	??
common/AException.hpp	??

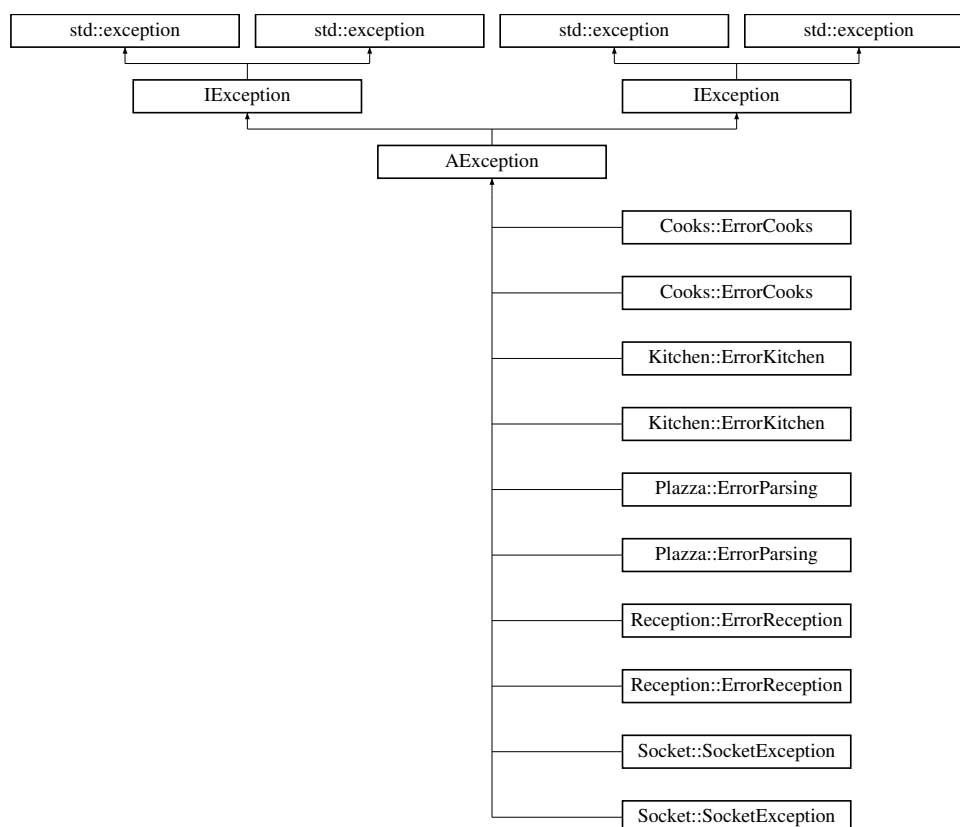
common/ APizza.hpp	??
common/ IException.hpp	??
common/ Ingridient.hpp	??
common/ IRecipe.hpp	??
common/ Socket.hpp	??
common/messages/ AStatus.hpp	??
common/messages/ CookStatus.hpp	??
common/messages/ DoneStatus.hpp	??
common/messages/ IMesagges.hpp	??
common/messages/ Inactivity.hpp	??
common/messages/ Order.hpp	??
common/messages/ Queue.hpp	??
common/messages/ RefillStatus.hpp	??
common/processes/ ForkProcess.hpp	??
common/processes/ IProcess.hpp	??
lib/ DLLoader.hpp	??
lib/ ILoader.hpp	??
src/ Plazza.hpp	??
src/cooks/ Cooks.hpp	??
src/kitchen/ Kitchen.hpp	??
src/reception/ Reception.hpp	??
src/recipes/pizza/ Americana.hpp	??
src/recipes/pizza/ Fantasia.hpp	??
src/recipes/pizza/ Margarita.hpp	??
src/recipes/pizza/ Regina.hpp	??
src/utls/ Utils.hpp	??

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`
- **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() [1/2]

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

Implements [IException](#).

5.1.1.2 getFormattedMessage() [2/2]

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

Implements [IException](#).

5.1.1.3 getMessage() [1/2]

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

Implements [IException](#).

5.1.1.4 getMessage() [2/2]

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

Implements [IException](#).

5.1.1.5 getType() [1/2]

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

Implements [IException](#).

5.1.1.6 `getType()` [2/2]

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

Implements [IException](#).

5.1.1.7 `what()` [1/2]

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

Implements [IException](#).

5.1.1.8 `what()` [2/2]

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

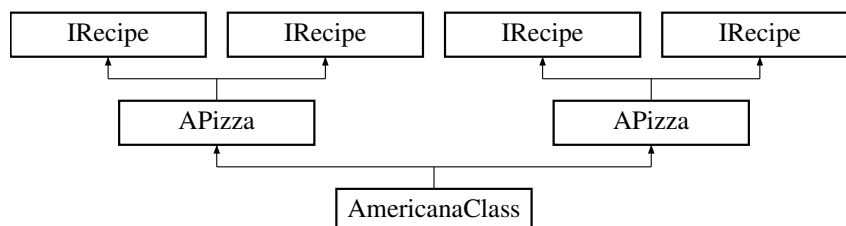
Implements [IException](#).

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

- bonus/common/AException.hpp
- 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
- **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](#)
- [APizza](#) ([int](#) number)
- [int](#) [getNumber](#) () [const override](#)
- [void](#) [setNumber](#) ([int](#) number) [override](#)

5.2.1 Member Function Documentation

5.2.1.1 [cook\(\)](#) [1/2]

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

Implements [APizza](#).

5.2.1.2 [cook\(\)](#) [2/2]

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

Implements [APizza](#).

5.2.1.3 [prepare\(\)](#) [1/2]

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

Implements [APizza](#).

5.2.1.4 [prepare\(\)](#) [2/2]

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

Implements [APizza](#).

5.2.1.5 [serve\(\)](#) [1/2]

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

Implements [APizza](#).

5.2.1.6 serve() [2/2]

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

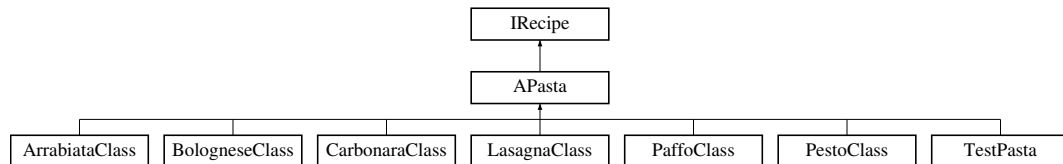
Implements [APizza](#).

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

- bonus/src/recipes/pizza/Americana.hpp
- src/recipes/pizza/Americana.hpp
- bonus/src/recipes/pizza/Americana.cpp
- 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 > ingradient](#)) [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]
```

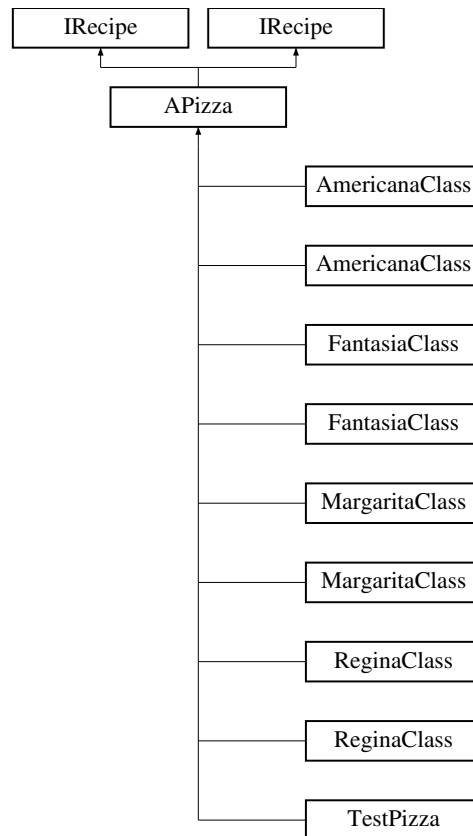
Implements [IRecipe](#).

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

- bonus/common/APasta.hpp
- bonus/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 > ingredient](#)) [override=0](#)
- [virtual void serve](#) () [override=0](#)
- [int](#) [getNumber](#) () [const](#) [override](#)
- [void](#) [setNumber](#) ([int](#) number) [override](#)
- **APizza** ([int](#) number)
- [virtual void cook](#) ([int](#) cookTime) [override=0](#)
- [virtual](#) [std::shared_ptr< Ingridient > prepare](#) ([int](#) number, [std::shared_ptr< Ingridient > ingredient](#)) [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() [1/2]

```

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

```

Implements [IRecipe](#).

5.4.1.2 cook() [2/2]

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

Implements [IRecipe](#).

5.4.1.3 getNumber() [1/2]

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

Implements [IRecipe](#).

5.4.1.4 getNumber() [2/2]

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

Implements [IRecipe](#).

5.4.1.5 prepare() [1/2]

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

Implements [IRecipe](#).

5.4.1.6 prepare() [2/2]

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

Implements [IRecipe](#).

5.4.1.7 serve() [1/2]

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

Implements [IRecipe](#).

5.4.1.8 serve() [2/2]

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

Implements [IRecipe](#).

5.4.1.9 `setNumber()` [1/2]

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

Implements [IRecipe](#).

5.4.1.10 `setNumber()` [2/2]

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

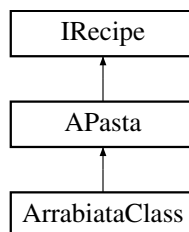
Implements [IRecipe](#).

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

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

5.5 ArrabiataClass Class Reference

Inheritance diagram for ArrabiataClass:



Public Member Functions

- **ArrabiataClass** ([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 ArrabiataClass::cook (
    int cookTime ) [override], [virtual]
```

Implements [APasta](#).

5.5.1.2 prepare()

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

Implements [APasta](#).

5.5.1.3 serve()

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

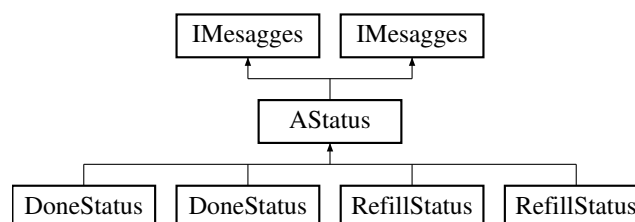
Implements [APasta](#).

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

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

5.6 AStatus Class Reference

Inheritance diagram for AStatus:



Public Member Functions

- **AStatus** (int id, std::vector< [ingStat](#) > status)
- std::string [pack](#) (const [IMesagges](#) &messages) const override
- virtual std::shared_ptr< [IMesagges](#) > [unpack](#) (const std::string &data) override=0
- std::string [typeToString](#) (MessageType type) const override
- virtual MessageType [getType](#) () const override=0
- **AStatus** (int id, std::vector< [ingStat](#) > status)
- std::string [pack](#) (const [IMesagges](#) &messages) const override
- virtual std::shared_ptr< [IMesagges](#) > [unpack](#) (const std::string &data) override=0
- std::string [typeToString](#) (MessageType type) const override
- virtual MessageType [getType](#) () const override=0

Public Attributes

- `int kitchenId`
- `std::vector< ingStat > status`

5.6.1 Member Function Documentation

5.6.1.1 `getType()` [1/2]

```
virtual MessageType AStatus::getType ( ) const [override], [pure virtual]
```

Implements [IMesaggess](#).

5.6.1.2 `getType()` [2/2]

```
virtual MessageType AStatus::getType ( ) const [override], [pure virtual]
```

Implements [IMesaggess](#).

5.6.1.3 `pack()` [1/2]

```
std::string AStatus::pack (
    const IMesaggess & messages ) const [override], [virtual]
```

Implements [IMesaggess](#).

5.6.1.4 `pack()` [2/2]

```
std::string AStatus::pack (
    const IMesaggess & messages ) const [override], [virtual]
```

Implements [IMesaggess](#).

5.6.1.5 `typeToString()` [1/2]

```
std::string AStatus::typeToString (
    MessageType type ) const [override], [virtual]
```

Implements [IMesaggess](#).

5.6.1.6 `typeToString()` [2/2]

```
std::string AStatus::typeToString (
    MessageType type ) const [override], [virtual]
```

Implements [IMesaggess](#).

5.6.1.7 unpack() [1/2]

```
virtual std::shared_ptr< IMessages > AStatus::unpack (
    const std::string & data ) [override], [pure virtual]
```

Implements [IMessages](#).

5.6.1.8 unpack() [2/2]

```
virtual std::shared_ptr< IMessages > AStatus::unpack (
    const std::string & data ) [override], [pure virtual]
```

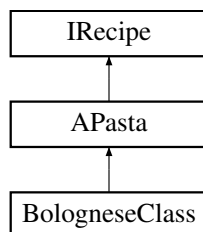
Implements [IMessages](#).

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

- bonus/common/messages/AStatus.hpp
- common/messages/AStatus.hpp
- bonus/common/messages/AStatus.cpp
- common/messages/AStatus.cpp

5.7 BologneseClass Class Reference

Inheritance diagram for BologneseClass:



Public Member Functions

- **BologneseClass** (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 BologneseClass::cook (
    int cookTime ) [override], [virtual]
```

Implements [APasta](#).

5.7.1.2 prepare()

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

Implements [APasta](#).

5.7.1.3 serve()

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

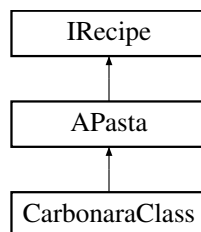
Implements [APasta](#).

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

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

5.8 CarbonaraClass Class Reference

Inheritance diagram for CarbonaraClass:



Public Member Functions

- **CarbonaraClass** ([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](#)

5.8.1 Member Function Documentation

5.8.1.1 cook()

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

Implements [APasta](#).

5.8.1.2 prepare()

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

Implements [APasta](#).

5.8.1.3 serve()

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

Implements [APasta](#).

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

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

5.9 Cooks Class Reference

Classes

- class [ErrorCooks](#)

Public Member Functions

- **Cooks** (std::shared_ptr< [Ingridient](#) > [ingradient](#), int id, int cookTime, int restockTime)
- std::shared_ptr< [Ingridient](#) > **startOrder** (std::shared_ptr< [Ingridient](#) > [ingradient](#), std::vector< std::string > [order](#))
- **bool hasEnoughIngredients** (const std::string &[orderData](#), std::shared_ptr< [Ingridient](#) > [ingradient](#))
- **void waitForTheOven** (Size size)
- std::shared_ptr< [IRecipe](#) > **loadPlugin** (const std::string &[path](#), int number)
- std::string **getType** (const std::string &[path](#), [DLLoader](#)< [IRecipe](#) > [loader](#))
- **int getID** () const
- **bool isBusy** () const
- **bool isRestocking** () const
- **void setIsBusy** (bool isBusy)
- std::string **toString** (RecipyType type)
- std::shared_ptr< [IRecipe](#) > **findAndLoadPlugin** (const std::string &[pizzaType](#), int number)
- **Cooks** (std::shared_ptr< [Ingridient](#) > [ingradient](#), int id, int cookTime, int restockTime)
- std::shared_ptr< [Ingridient](#) > **startOrder** (std::shared_ptr< [Ingridient](#) > [ingradient](#), std::vector< std::string > [order](#))
- **bool hasEnoughIngredients** (const std::string &[orderData](#), std::shared_ptr< [Ingridient](#) > [ingradient](#))
- std::shared_ptr< [IRecipe](#) > **loadPlugin** (const std::string &[path](#), int number)
- std::string **getType** (const std::string &[path](#), [DLLoader](#)< [IRecipe](#) > [loader](#))
- **int getID** () const
- **bool isBusy** () const
- **bool isRestocking** () const
- **void setIsBusy** (bool isBusy)
- std::string **toString** (PizzaType type)
- std::shared_ptr< [IRecipe](#) > **findAndLoadPlugin** (const std::string &[pizzaType](#), int number)

Private Attributes

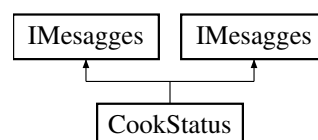
- [int](#) _ID
- [int](#) _cookTime
- [int](#) _restockTime
- std::shared_ptr< [Ingridient](#) > _ingradient
- std::mutex _statusMutex
- [bool](#) _isBusy
- [bool](#) _isRestocking

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

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

5.10 CookStatus Class Reference

Inheritance diagram for CookStatus:



Public Member Functions

- **CookStatus** (int kitchenID, const std::vector< [CookStatusData](#) > &cooksStatus)
- std::string [pack](#) (const [IMesaggess](#) &messages) const override
- std::shared_ptr< [IMesaggess](#) > [unpack](#) (const std::string &data) override
- MessageType [getType](#) () const override
- std::string [typeToString](#) (MessageType type) const override
- **CookStatus** (int kitchenID, const std::vector< [CookStatusData](#) > &cooksStatus)
- std::string [pack](#) (const [IMesaggess](#) &messages) const override
- std::shared_ptr< [IMesaggess](#) > [unpack](#) (const std::string &data) override
- MessageType [getType](#) () const override
- std::string [typeToString](#) (MessageType type) const override

Public Attributes

- int [_kitchenId](#)
- std::vector< [CookStatusData](#) > [_cooksStatus](#)

5.10.1 Member Function Documentation

5.10.1.1 [getType\(\)](#) [1/2]

```
MessageType CookStatus::getType ( ) const [override], [virtual]
```

Implements [IMesaggess](#).

5.10.1.2 [getType\(\)](#) [2/2]

```
MessageType CookStatus::getType ( ) const [override], [virtual]
```

Implements [IMesaggess](#).

5.10.1.3 [pack\(\)](#) [1/2]

```
std::string CookStatus::pack (
    const IMesaggess & messages ) const [override], [virtual]
```

Implements [IMesaggess](#).

5.10.1.4 [pack\(\)](#) [2/2]

```
std::string CookStatus::pack (
    const IMesaggess & messages ) const [override], [virtual]
```

Implements [IMesaggess](#).

5.10.1.5 typeToString() [1/2]

```
std::string CookStatus::typeToString (
    MessageType type ) const [override], [virtual]
```

Implements [IMesagges](#).

5.10.1.6 typeToString() [2/2]

```
std::string CookStatus::typeToString (
    MessageType type ) const [override], [virtual]
```

Implements [IMesagges](#).

5.10.1.7 unpack() [1/2]

```
std::shared_ptr< IMesagges > CookStatus::unpack (
    const std::string & data ) [override], [virtual]
```

Implements [IMesagges](#).

5.10.1.8 unpack() [2/2]

```
std::shared_ptr< IMesagges > CookStatus::unpack (
    const std::string & data ) [override], [virtual]
```

Implements [IMesagges](#).

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

- bonus/common/messages/CookStatus.hpp
- common/messages/CookStatus.hpp
- bonus/common/messages/CookStatus.cpp
- common/messages/CookStatus.cpp

5.11 CookStatusData Struct Reference

Public Attributes

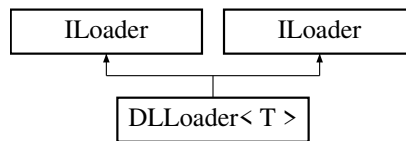
- [int](#) cookId
- [bool](#) isBusy
- [bool](#) isRestocking

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

- bonus/common/messages/CookStatus.hpp
- common/messages/CookStatus.hpp

5.12 DLLoader< T > Class Template Reference

Inheritance diagram for DLLoader< T >:



Public Member Functions

- `void * getHandler () const override`
- `void * Open (const char *path, int flag) override`
- `void * Symbol (const char *symbolName) override`
- `T getSymbol (const char *symbolName)`
- `int Close () override`
- `const char * Error () override`
- `void * getHandler () const override`
- `void * Open (const char *path, int flag) override`
- `void * Symbol (const char *symbolName) override`
- `T getSymbol (const char *symbolName)`
- `int Close () override`
- `const char * Error () override`

Private Attributes

- `void * _handler = nullptr`

5.12.1 Member Function Documentation

5.12.1.1 Close() [1/2]

```
template<typename T >
int DLLoader< T >::Close ( ) [inline], [override], [virtual]
```

Implements [ILoader](#).

5.12.1.2 Close() [2/2]

```
template<typename T >
int DLLoader< T >::Close ( ) [inline], [override], [virtual]
```

Implements [ILoader](#).

5.12.1.3 Error() [1/2]

```
template<typename T >
const char * DLLoader< T >::Error ( ) [inline], [override], [virtual]
```

Implements [ILoader](#).

5.12.1.4 Error() [2/2]

```
template<typename T >
const char * DLLoader< T >::Error ( ) [inline], [override], [virtual]
```

Implements [ILoader](#).

5.12.1.5 getHandler() [1/2]

```
template<typename T >
void * DLLoader< T >::getHandler ( ) const [inline], [override], [virtual]
```

Implements [ILoader](#).

5.12.1.6 getHandler() [2/2]

```
template<typename T >
void * DLLoader< T >::getHandler ( ) const [inline], [override], [virtual]
```

Implements [ILoader](#).

5.12.1.7 Open() [1/2]

```
template<typename T >
void * DLLoader< T >::Open (
    const char * path,
    int flag ) [inline], [override], [virtual]
```

Implements [ILoader](#).

5.12.1.8 Open() [2/2]

```
template<typename T >
void * DLLoader< T >::Open (
    const char * path,
    int flag ) [inline], [override], [virtual]
```

Implements [ILoader](#).

5.12.1.9 Symbol() [1/2]

```
template<typename T >
void * DLoader< T >::Symbol (
    const char * symbolName ) [inline], [override], [virtual]
```

Implements [ILoader](#).

5.12.1.10 Symbol() [2/2]

```
template<typename T >
void * DLoader< T >::Symbol (
    const char * symbolName ) [inline], [override], [virtual]
```

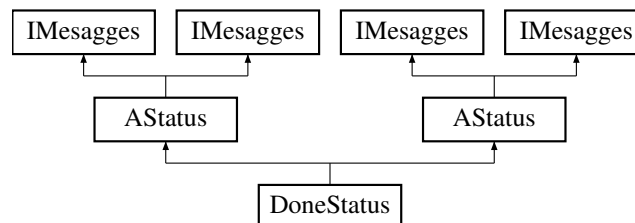
Implements [ILoader](#).

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

- bonus/lib/DLoader.hpp
- lib/DLoader.hpp

5.13 DoneStatus Class Reference

Inheritance diagram for DoneStatus:



Public Member Functions

- **DoneStatus** (int id, std::vector< [ingStat](#) > status)
- MessageType [getType](#) () [const override](#)
- std::shared_ptr< [IMesagges](#) > [unpack](#) (const std::string &data) [override](#)
- **DoneStatus** (int id, std::vector< [ingStat](#) > status)
- MessageType [getType](#) () [const override](#)
- std::shared_ptr< [IMesagges](#) > [unpack](#) (const std::string &data) [override](#)

Public Member Functions inherited from [AStatus](#)

- **AStatus** (int id, std::vector< [ingStat](#) > status)
- std::string [pack](#) (const [IMesagges](#) &messages) [const override](#)
- std::string [typeToString](#) (MessageType type) [const override](#)
- **AStatus** (int id, std::vector< [ingStat](#) > status)
- std::string [pack](#) (const [IMesagges](#) &messages) [const override](#)
- std::string [typeToString](#) (MessageType type) [const override](#)

Additional Inherited Members

Public Attributes inherited from [AStatus](#)

- [int](#) kitchenId
- std::vector< [ingStat](#) > status

5.13.1 Member Function Documentation

5.13.1.1 `getType()` [1/2]

```
MessageType DoneStatus::getType ( ) const [override], [virtual]
```

Implements [AStatus](#).

5.13.1.2 `getType()` [2/2]

```
MessageType DoneStatus::getType ( ) const [override], [virtual]
```

Implements [AStatus](#).

5.13.1.3 `unpack()` [1/2]

```
std::shared_ptr< IMesagges > DoneStatus::unpack (
    const std::string & data ) [override], [virtual]
```

Implements [AStatus](#).

5.13.1.4 `unpack()` [2/2]

```
std::shared_ptr< IMesagges > DoneStatus::unpack (
    const std::string & data ) [override], [virtual]
```

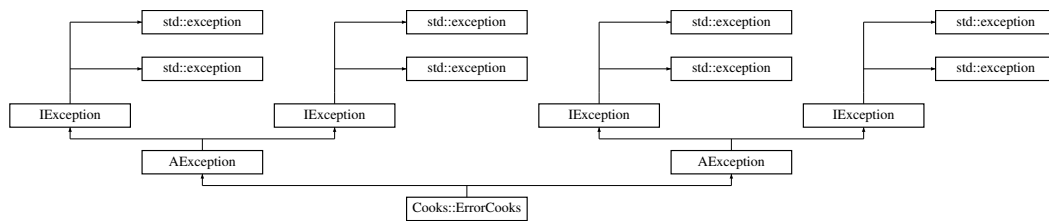
Implements [AStatus](#).

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

- bonus/common/messages/DoneStatus.hpp
- common/messages/DoneStatus.hpp
- bonus/common/messages/DoneStatus.cpp
- common/messages/DoneStatus.cpp

5.14 Cooks::ErrorCooks Class Reference

Inheritance diagram for Cooks::ErrorCooks:



Public Member Functions

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

Public Member Functions inherited from AException

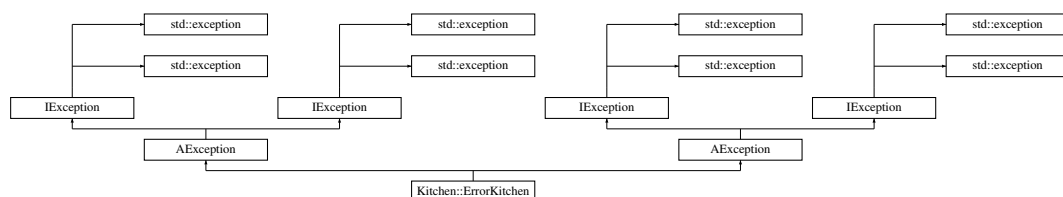
- **AException** (`const std::string &type, const std::string &message`)
- `const char * what () const noexcept override`
- `std::string getType () const noexcept override`
- `std::string getMessage () const noexcept override`
- `std::string getFormattedMessage () const noexcept override`
- **AException**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

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

5.15 Kitchen::ErrorKitchen Class Reference

Inheritance diagram for Kitchen::ErrorKitchen:



Public Member Functions

- **ErrorKitchen** ([const](#) std::string &[message](#))
- **ErrorKitchen** ([const](#) std::string &[message](#))

Public Member Functions inherited from [AException](#)

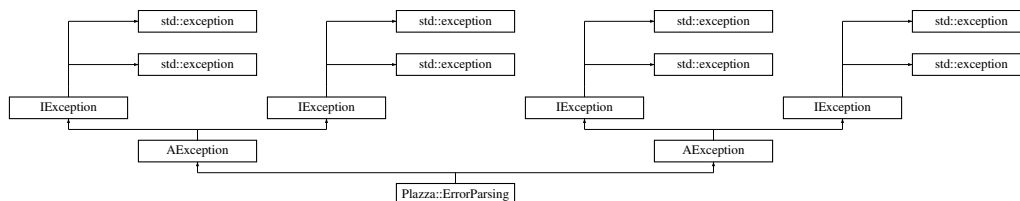
- **AException** ([const](#) std::string &[type](#), [const](#) std::string &[message](#))
- [const](#) char * [what](#) () [const](#) noexcept [override](#)
- std::string [getType](#) () [const](#) noexcept [override](#)
- std::string [getMessage](#) () [const](#) noexcept [override](#)
- std::string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

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

5.16 Plaza::ErrorParsing Class Reference

Inheritance diagram for Plaza::ErrorParsing:



Public Member Functions

- **ErrorParsing** ([const](#) std::string &[message](#))
- **ErrorParsing** ([const](#) std::string &[message](#))

Public Member Functions inherited from **AException**

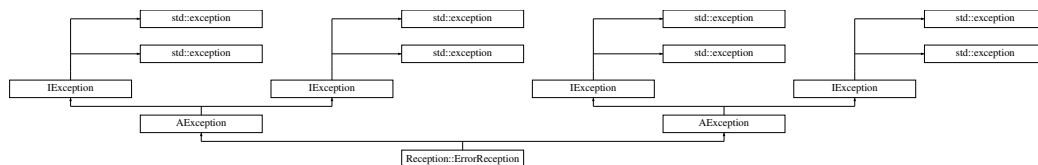
- **AException** (`const std::string &type`, `const std::string &message`)
- `const char * what () const noexcept override`
- `std::string getType () const noexcept override`
- `std::string getMessage () const noexcept override`
- `std::string getFormattedMessage () const noexcept override`
- **AException**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

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

5.17 Reception::ErrorReception Class Reference

Inheritance diagram for Reception::ErrorReception:



Public Member Functions

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

Public Member Functions inherited from **AException**

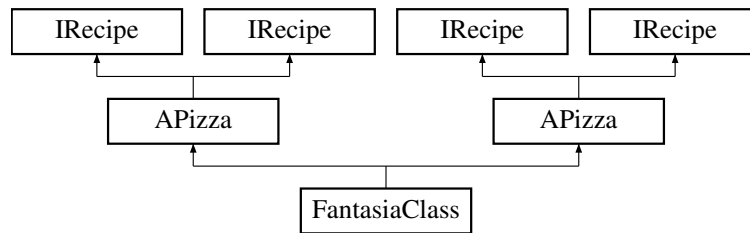
- **AException** (`const std::string &type`, `const std::string &message`)
- `const char * what () const noexcept override`
- `std::string getType () const noexcept override`
- `std::string getMessage () const noexcept override`
- `std::string getFormattedMessage () const noexcept override`
- **AException**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

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

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

Public Member Functions inherited from APizza

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

5.18.1 Member Function Documentation

5.18.1.1 cook() [1/2]

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

Implements [APizza](#).

5.18.1.2 cook() [2/2]

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

Implements [APizza](#).

5.18.1.3 prepare() [1/2]

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

Implements [APizza](#).

5.18.1.4 prepare() [2/2]

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

Implements [APizza](#).

5.18.1.5 serve() [1/2]

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

Implements [APizza](#).

5.18.1.6 serve() [2/2]

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

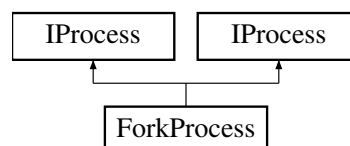
Implements [APizza](#).

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

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

5.19 ForkProcess Class Reference

Inheritance diagram for ForkProcess:

**Classes**

- class [ProcessException](#)

Public Member Functions

- `pid_t create (const std::function< void()> &childLogic) override`
- `int wait () override`
- `void close () override`
- `pid_t getPid () const override`
- `pid_t create (const std::function< void()> &childLogic) override`
- `int wait () override`
- `void close () override`
- `pid_t getPid () const override`

Private Attributes

- `pid_t _pid`

5.19.1 Member Function Documentation

5.19.1.1 close() [1/2]

```
void ForkProcess::close ( ) [override], [virtual]
```

Implements [IProcess](#).

5.19.1.2 close() [2/2]

```
void ForkProcess::close ( ) [override], [virtual]
```

Implements [IProcess](#).

5.19.1.3 create() [1/2]

```
pid_t ForkProcess::create (
    const std::function< void()> & childLogic ) [override], [virtual]
```

Implements [IProcess](#).

5.19.1.4 create() [2/2]

```
pid_t ForkProcess::create (
    const std::function< void()> & childLogic ) [override], [virtual]
```

Implements [IProcess](#).

5.19.1.5 getPid() [1/2]

```
pid_t ForkProcess::getPid ( ) const [override], [virtual]
```

Implements [IProcess](#).

5.19.1.6 getpid() [2/2]

```
pid_t ForkProcess::getPid ( ) const [override], [virtual]
```

Implements [IProcess](#).

5.19.1.7 wait() [1/2]

```
int ForkProcess::wait ( ) [override], [virtual]
```

Implements [IProcess](#).

5.19.1.8 wait() [2/2]

```
int ForkProcess::wait ( ) [override], [virtual]
```

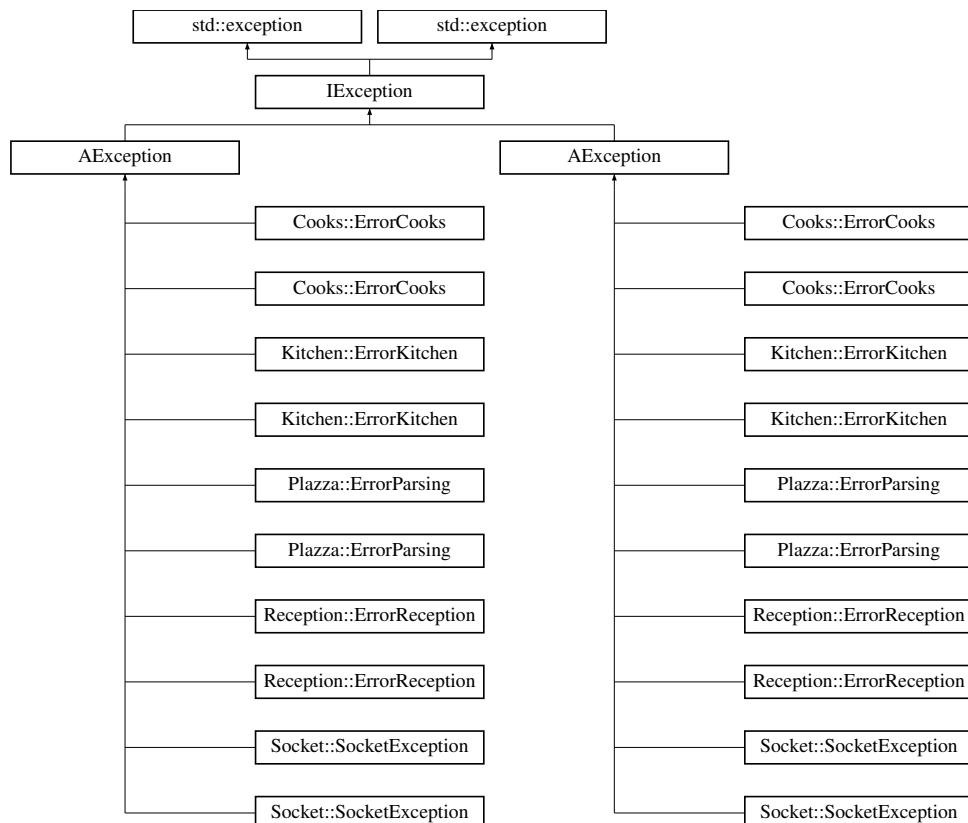
Implements [IProcess](#).

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

- bonus/common/processes/ForkProcess.hpp
- common/processes/ForkProcess.hpp
- bonus/common/processes/ForkProcess.cpp
- common/processes/ForkProcess.cpp

5.20 IException Class Reference

Inheritance diagram for IException:



Public Member Functions

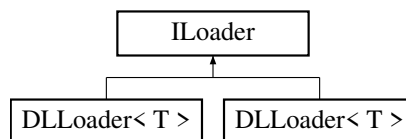
- `const char * what () const noexcept override=0`
- `virtual std::string getType () const noexcept=0`
- `virtual std::string getMessage () const noexcept=0`
- `virtual std::string getFormattedMessage () const noexcept=0`
- `const char * what () const noexcept override=0`
- `virtual std::string getType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

- `bonus/common/IException.hpp`
- `common/IException.hpp`

5.21 ILoader Class Reference

Inheritance diagram for ILoader:



Public Member Functions

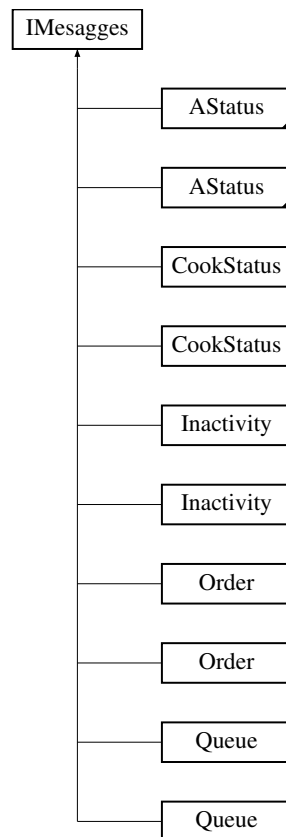
- `virtual void * Open (const char *path, int flag)=0`
- `virtual void * Symbol (const char *symbolName)=0`
- `virtual int Close ()=0`
- `virtual const char * Error ()=0`
- `virtual void * getHandler () const =0`
- `virtual void * Open (const char *path, int flag)=0`
- `virtual void * Symbol (const char *symbolName)=0`
- `virtual int Close ()=0`
- `virtual const char * Error ()=0`
- `virtual void * getHandler () const =0`

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

- `bonus/lib/ILoader.hpp`
- `lib/ILoader.hpp`

5.22 IMessages Class Reference

Inheritance diagram for IMessages:



Public Member Functions

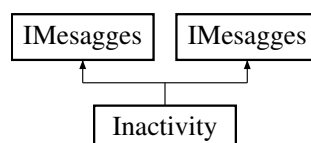
- `virtual std::string pack (const IMessages &messages) const =0`
- `virtual std::shared_ptr< IMessages > unpack (const std::string &data)=0`
- `virtual MessageType getType () const =0`
- `virtual std::string typeToString (MessageType type) const =0`
- `virtual std::string pack (const IMessages &messages) const =0`
- `virtual std::shared_ptr< IMessages > unpack (const std::string &data)=0`
- `virtual MessageType getType () const =0`
- `virtual std::string typeToString (MessageType type) const =0`

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

- `bonus/common/messages/IMessages.hpp`
- `common/messages/IMessages.hpp`

5.23 Inactivity Class Reference

Inheritance diagram for Inactivity:



Public Member Functions

- **Inactivity** ([int](#) kitchenId)
- [std::string](#) [pack](#) ([const IMesagges](#) &messages) [const override](#)
- [std::shared_ptr< IMesagges >](#) [unpack](#) ([const std::string](#) &data) [override](#)
- [MessageType](#) [getType](#) () [const override](#)
- [std::string](#) [typeToString](#) ([MessageType](#) type) [const override](#)
- **Inactivity** ([int](#) kitchenId)
- [std::string](#) [pack](#) ([const IMesagges](#) &messages) [const override](#)
- [std::shared_ptr< IMesagges >](#) [unpack](#) ([const std::string](#) &data) [override](#)
- [MessageType](#) [getType](#) () [const override](#)
- [std::string](#) [typeToString](#) ([MessageType](#) type) [const override](#)

Public Attributes

- [int](#) id

5.23.1 Member Function Documentation

5.23.1.1 [getType\(\)](#) [1/2]

```
MessageType Inactivity::getType ( ) const [override], [virtual]
```

Implements [IMesagges](#).

5.23.1.2 [getType\(\)](#) [2/2]

```
MessageType Inactivity::getType ( ) const [override], [virtual]
```

Implements [IMesagges](#).

5.23.1.3 [pack\(\)](#) [1/2]

```
std::string Inactivity::pack (
    const IMesagges & messages ) const [override], [virtual]
```

Implements [IMesagges](#).

5.23.1.4 [pack\(\)](#) [2/2]

```
std::string Inactivity::pack (
    const IMesagges & messages ) const [override], [virtual]
```

Implements [IMesagges](#).

5.23.1.5 typeToString() [1/2]

```
std::string Inactivity::typeToString (
    MessageType type ) const [override], [virtual]
```

Implements [IMesagges](#).

5.23.1.6 typeToString() [2/2]

```
std::string Inactivity::typeToString (
    MessageType type ) const [override], [virtual]
```

Implements [IMesagges](#).

5.23.1.7 unpack() [1/2]

```
std::shared_ptr< IMesagges > Inactivity::unpack (
    const std::string & data ) [override], [virtual]
```

Implements [IMesagges](#).

5.23.1.8 unpack() [2/2]

```
std::shared_ptr< IMesagges > Inactivity::unpack (
    const std::string & data ) [override], [virtual]
```

Implements [IMesagges](#).

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

- bonus/common/messages/Inactivity.hpp
- common/messages/Inactivity.hpp
- bonus/common/messages/Inactivity.cpp
- common/messages/Inactivity.cpp

5.24 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`
- `std::shared_ptr< Ingridient > operator= (const std::vector< ingStat > &ingStat)`
- `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`
- `std::shared_ptr< Ingridient > operator= (const std::vector< ingStat > &ingStat)`

Static Public Member Functions

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

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

- `bonus/common/Ingradient.hpp`
- `common/Ingradient.hpp`
- `bonus/common/Ingradient.cpp`
- `common/Ingradient.cpp`

5.25 `ingStat` Struct Reference

Public Attributes

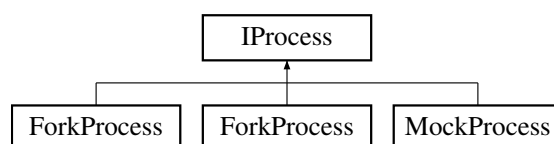
- `IngradientType type`
- `int quantity`

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

- `bonus/common/Ingradient.hpp`
- `common/Ingradient.hpp`

5.26 `IProcess` Class Reference

Inheritance diagram for `IProcess`:



Public Member Functions

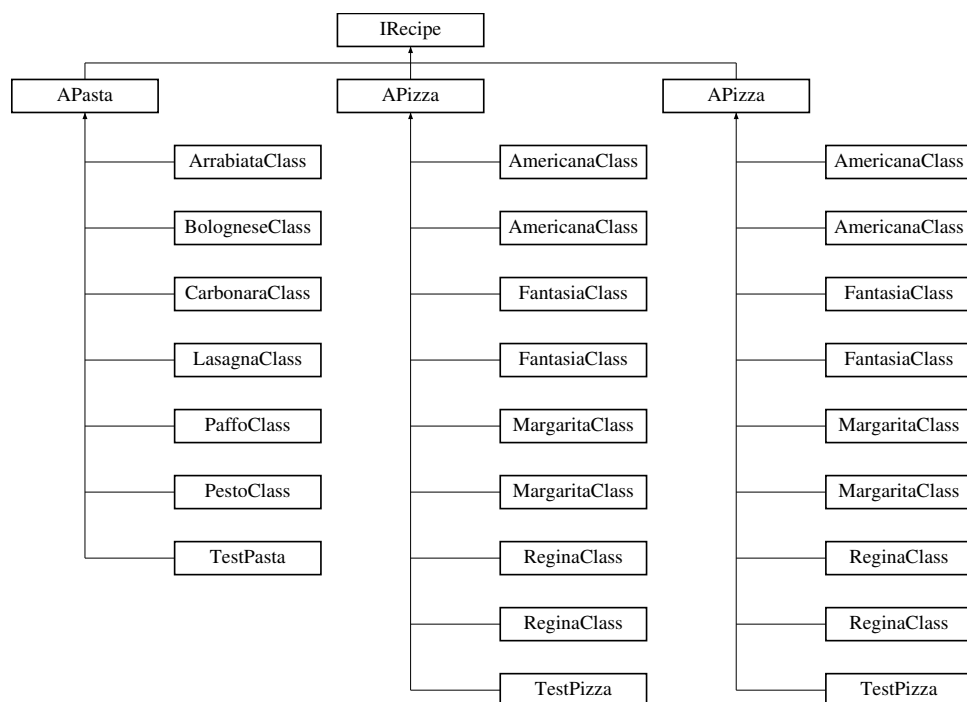
- `virtual pid_t create (const std::function< void()> &childLogic)=0`
- `virtual int wait ()=0`
- `virtual void close ()=0`
- `virtual pid_t getPid () const =0`
- `virtual pid_t create (const std::function< void()> &childLogic)=0`
- `virtual int wait ()=0`
- `virtual void close ()=0`
- `virtual pid_t getPid () const =0`

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

- `bonus/common/processes/IProcess.hpp`
- `common/processes/IProcess.hpp`

5.27 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`
- `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 files:

- `bonus/common/IRecipe.hpp`
- `common/IRecipe.hpp`

5.28 Kitchen Class Reference

Classes

- class `ErrorKitchen`

Public Member Functions

- `Kitchen (int id, int nbCooks, int cookTime, int restockTime, bool debug)`
- `void restock ()`
- `void startKitchenProcess ()`
- `void startKitchen ()`
- `void run ()`
- `void processOrder (const std::string &orderData)`
- `bool canAcceptOrder (int numPizzas)`
- `void stopKitchen ()`
- `void sendOrder ()`
- `void createCooks ()`
- `void setIsFull (bool isFull)`
- `void setProcess (std::shared_ptr< IProcess > process)`
- `void setCurrentOrders (int currentOrders)`
- `void incrementCurrentOrders (int amount)`
- `int getID () const`
- `int getNbCooks () const`
- `int getCookTime () const`
- `int getRestockTime () const`
- `int getMaxCmd () const`
- `std::shared_ptr< Ingridient > getIngridient () const`
- `std::vector< std::shared_ptr< Cooks > > getCooks () const`
- `int getCurrentOrders () const`
- `bool isFull () const`
- `void sendQueueStatMessage ()`
- `void sendDoneMessage ()`
- `void sendRefillMessage ()`
- `void sendInactive ()`
- `void sendCookStatus ()`
- `Kitchen (int id, int nbCooks, int cookTime, int restockTime, bool debug)`
- `void restock ()`
- `void startKitchenProcess ()`
- `void startKitchen ()`
- `void run ()`

- **void processOrder** ([const](#) std::string &[orderData](#))
- **bool canAcceptOrder** ([int](#) numPizzas)
- **void stopKitchen** ()
- **void sendOrder** ()
- **void createCooks** ()
- **void setIsFull** ([bool](#) isFull)
- **void setProcess** (std::shared_ptr< [IProcess](#) > [process](#))
- **void setCurrentOrders** ([int](#) currentOrders)
- **void incrementCurrentOrders** ([int](#) amount)
- **int getID** () [const](#)
- **int getNbCooks** () [const](#)
- **int getCookTime** () [const](#)
- **int getRestockTime** () [const](#)
- **int getMaxCmd** () [const](#)
- std::shared_ptr< [Ingridient](#) > **getIngridient** () [const](#)
- std::vector< std::shared_ptr< [Cooks](#) > > **getCooks** () [const](#)
- **int getCurrentOrders** () [const](#)
- **bool isFull** () [const](#)
- **void sendQueueStatMessage** ()
- **void sendDoneMessage** ()
- **void sendRefillMessage** ()
- **void sendInactive** ()
- **void sendCookStatus** ()

Private Attributes

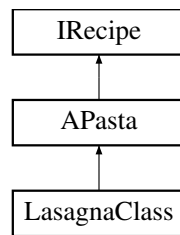
- [int](#) _ID
- [int](#) _nbCooks
- [int](#) _cookTime
- [int](#) _restockTime
- [int](#) _maxCmd
- [int](#) _currentOrders
- [pid_t](#) _pid
- std::shared_ptr< [Ingridient](#) > _ingridient
- std::vector< std::shared_ptr< [Cooks](#) > > _cooks
- std::queue< std::string > _orderQueue
- std::mutex _orderMutex
- std::mutex _ingMutex
- std::condition_variable _cookCV
- std::vector< std::thread > _cookThreads
- std::thread _restockThread
- [Socket](#) _socket
- std::chrono::steady_clock::time_point _lastActivity
- std::shared_ptr< [IProcess](#) > _process
- [bool](#) _isRunning
- [bool](#) _isDebug
- [bool](#) _isFull

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

- bonus/src/kitchen/Kitchen.hpp
- src/kitchen/Kitchen.hpp
- bonus/src/kitchen/Kitchen.cpp
- bonus/src/kitchen/SendingMessageKitchen.cpp
- src/kitchen/Kitchen.cpp
- src/kitchen/SendingMessageKitchen.cpp
- temp.cpp

5.29 LasagnaClass Class Reference

Inheritance diagram for LasagnaClass:



Public Member Functions

- **LasagnaClass** ([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](#)

5.29.1 Member Function Documentation

5.29.1.1 cook()

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

Implements [APasta](#).

5.29.1.2 prepare()

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

Implements [APasta](#).

5.29.1.3 serve()

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

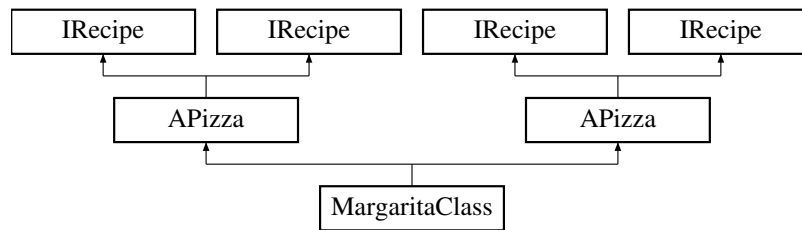
Implements [APasta](#).

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

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

5.30 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 > ingradient](#)) [override](#)
- [void serve](#) () [override](#)
- **MargaritaClass** ([int](#) number)
- [void cook](#) ([int](#) cookTime) [override](#)
- [std::shared_ptr< Ingridient > prepare](#) ([int](#) number, [std::shared_ptr< Ingridient > ingradient](#)) [override](#)
- [void serve](#) () [override](#)

Public Member Functions inherited from [APizza](#)

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

5.30.1 Member Function Documentation

5.30.1.1 [cook\(\)](#) [1/2]

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

Implements [APizza](#).

5.30.1.2 [cook\(\)](#) [2/2]

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

Implements [APizza](#).

5.30.1.3 prepare() [1/2]

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

Implements [APizza](#).

5.30.1.4 prepare() [2/2]

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

Implements [APizza](#).

5.30.1.5 serve() [1/2]

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

Implements [APizza](#).

5.30.1.6 serve() [2/2]

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

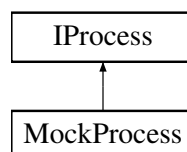
Implements [APizza](#).

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

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

5.31 MockProcess Class Reference

Inheritance diagram for MockProcess:



Public Member Functions

- **MockProcess** ([bool shouldFailCreate=false](#), [bool shouldFailClose=false](#))
- [pid_t](#) **getPid** () [const override](#)
- [pid_t](#) **create** ([const std::function< void\(\)> &childLogic](#)) [override](#)
- [void](#) **close** () [override](#)
- [int](#) **wait** () [override](#)
- [void](#) **executeStoredFunction** ()
- [void](#) **setMockPid** ([pid_t](#) pid)
- [void](#) **setShouldFailCreate** ([bool](#) fail)
- [void](#) **setShouldFailClose** ([bool](#) fail)

Private Attributes

- [bool](#) **_shouldFailCreate**
- [bool](#) **_shouldFailClose**
- [pid_t](#) **_mockPid**
- [std::function< void\(\)>](#) **_storedFunc**

5.31.1 Member Function Documentation

5.31.1.1 close()

```
void MockProcess::close ( ) [inline], [override], [virtual]
```

Implements [IProcess](#).

5.31.1.2 create()

```
pid\_t MockProcess::create (
    const std::function< void\(\)> & childLogic ) [inline], [override], [virtual]
```

Implements [IProcess](#).

5.31.1.3 getPid()

```
pid\_t MockProcess::getPid ( ) const [inline], [override], [virtual]
```

Implements [IProcess](#).

5.31.1.4 wait()

```
int MockProcess::wait ( ) [inline], [override], [virtual]
```

Implements [IProcess](#).

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

- tests/Kitchen-test.cpp

5.32 MockSocket Class Reference

Public Member Functions

- `void createServer (const std::string &path)`
- `void acceptClient ()`
- `void connectToServer (const std::string &path)`
- `bool isConnected () const`
- `void send (const std::string &msg)`
- `std::string receive ()`
- `void addMockMessage (const std::string &msg)`
- `void disconnect ()`

Public Attributes

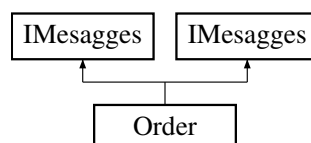
- `bool _shouldFailCreate`
- `bool _shouldFailAccept`
- `bool _shouldFailConnect`
- `bool _isConnected`
- `std::string _lastSentMessage`
- `std::queue< std::string > _receivedMessages`

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

- `tests/Kitchen-test.cpp`

5.33 Order Class Reference

Inheritance diagram for Order:



Public Member Functions

- `Order (RecipyType t, Size s, int n)`
- `std::string pack (const IMesagges &order) const override`
- `std::shared_ptr< IMesagges > unpack (const std::string &data) override`
- `MessageType getType () const override`
- `std::string typeToString (MessageType type) const override`
- `Order (PizzaType t, Size s, int n)`
- `std::string pack (const IMesagges &order) const override`
- `std::shared_ptr< IMesagges > unpack (const std::string &data) override`
- `MessageType getType () const override`
- `std::string typeToString (MessageType type) const override`

Public Attributes

- RecipyType **type**
- Size **size**
- [int](#) **number**
- PizzaType **type**

5.33.1 Member Function Documentation

5.33.1.1 `getType()` [1/2]

```
MessageType Order::getType ( ) const [override], [virtual]
```

Implements [IMesaggess](#).

5.33.1.2 `getType()` [2/2]

```
MessageType Order::getType ( ) const [override], [virtual]
```

Implements [IMesaggess](#).

5.33.1.3 `pack()` [1/2]

```
std::string Order::pack (
    const IMesaggess & order ) const [override], [virtual]
```

Implements [IMesaggess](#).

5.33.1.4 `pack()` [2/2]

```
std::string Order::pack (
    const IMesaggess & order ) const [override], [virtual]
```

Implements [IMesaggess](#).

5.33.1.5 `typeToString()` [1/2]

```
std::string Order::typeToString (
    MessageType type ) const [override], [virtual]
```

Implements [IMesaggess](#).

5.33.1.6 `typeToString()` [2/2]

```
std::string Order::typeToString (
    MessageType type ) const [override], [virtual]
```

Implements [IMesaggess](#).

5.33.1.7 unpack() [1/2]

```
std::shared_ptr< IMessages > Order::unpack (
    const std::string & data ) [override], [virtual]
```

Implements [IMessages](#).

5.33.1.8 unpack() [2/2]

```
std::shared_ptr< IMessages > Order::unpack (
    const std::string & data ) [override], [virtual]
```

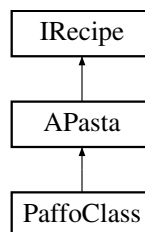
Implements [IMessages](#).

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

- bonus/common/messages/Order.hpp
- common/messages/Order.hpp
- bonus/common/messages/Order.cpp
- common/messages/Order.cpp

5.34 PaffoClass Class Reference

Inheritance diagram for PaffoClass:



Public Member Functions

- **PaffoClass** (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

5.34.1 Member Function Documentation

5.34.1.1 cook()

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

Implements [APasta](#).

5.34.1.2 prepare()

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

Implements [APasta](#).

5.34.1.3 serve()

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

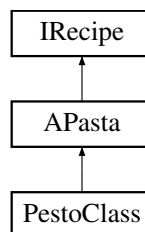
Implements [APasta](#).

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

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

5.35 PestoClass Class Reference

Inheritance diagram for PestoClass:



Public Member Functions

- **PestoClass** (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](#)

5.35.1 Member Function Documentation

5.35.1.1 [cook\(\)](#)

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

Implements [APasta](#).

5.35.1.2 [prepare\(\)](#)

```
std::shared\_ptr< Ingridient > PestoClass::prepare (  
    int number,  
    std::shared\_ptr< Ingridient > ingradient )    [override], [virtual]
```

Implements [APasta](#).

5.35.1.3 [serve\(\)](#)

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

Implements [APasta](#).

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

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

5.36 Plaza Class Reference

Classes

- class [ErrorParsing](#)

Public Member Functions

- [void](#) [parseCmd](#) ([char](#) **[av](#), [int](#) [ac](#))
- [void](#) [orderingLoop](#) ()
- [void](#) [parseCmd](#) ([char](#) **[av](#), [int](#) [ac](#))
- [void](#) [orderingLoop](#) ()

Private Attributes

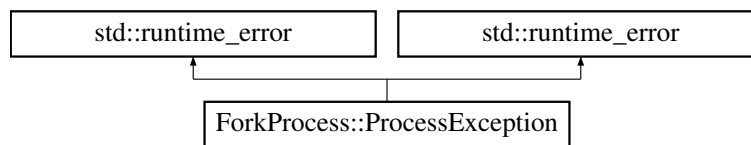
- [int](#) `_nbCooks`
- [int](#) `_timerCooker`
- [int](#) `_timerRestock`
- [bool](#) `_debug`
- [Reception](#) `_reception`

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

- `bonus/src/Plazza.hpp`
- `src/Plazza.hpp`
- `bonus/src/Plazza.cpp`
- `src/Plazza.cpp`

5.37 ForkProcess::ProcessException Class Reference

Inheritance diagram for ForkProcess::ProcessException:



Public Member Functions

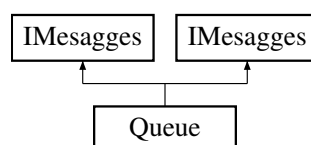
- **ProcessException** ([const](#) `std::string` &[msg](#))
- **ProcessException** ([const](#) `std::string` &[msg](#))

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

- `bonus/common/processes/ForkProcess.hpp`
- `common/processes/ForkProcess.hpp`

5.38 Queue Class Reference

Inheritance diagram for Queue:



Public Member Functions

- **Queue** ([int](#) id, [int](#) currentOrders)
- [std::string](#) **pack** ([const](#) [IMesaggex](#) &messages) [const override](#)
- [std::shared_ptr](#)< [IMesaggex](#) > **unpack** ([const](#) [std::string](#) &data) [override](#)
- [MessageType](#) **getType** () [const override](#)
- [std::string](#) **typeToString** ([MessageType](#) type) [const override](#)
- **Queue** ([int](#) id, [int](#) currentOrders)
- [std::string](#) **pack** ([const](#) [IMesaggex](#) &messages) [const override](#)
- [std::shared_ptr](#)< [IMesaggex](#) > **unpack** ([const](#) [std::string](#) &data) [override](#)
- [MessageType](#) **getType** () [const override](#)
- [std::string](#) **typeToString** ([MessageType](#) type) [const override](#)

Public Attributes

- [int](#) kitchenId
- [int](#) nbCurrentOrders

5.38.1 Member Function Documentation

5.38.1.1 **getType()** [1/2]

```
MessageType Queue::getType ( ) const [override], [virtual]
```

Implements [IMesaggex](#).

5.38.1.2 **getType()** [2/2]

```
MessageType Queue::getType ( ) const [override], [virtual]
```

Implements [IMesaggex](#).

5.38.1.3 **pack()** [1/2]

```
std::string Queue::pack (
    const IMesaggex & messages ) const [override], [virtual]
```

Implements [IMesaggex](#).

5.38.1.4 **pack()** [2/2]

```
std::string Queue::pack (
    const IMesaggex & messages ) const [override], [virtual]
```

Implements [IMesaggex](#).

5.38.1.5 typeToString() [1/2]

```
std::string Queue::typeToString (
    MessageType type ) const [override], [virtual]
```

Implements [IMesagges](#).

5.38.1.6 typeToString() [2/2]

```
std::string Queue::typeToString (
    MessageType type ) const [override], [virtual]
```

Implements [IMesagges](#).

5.38.1.7 unpack() [1/2]

```
std::shared_ptr< IMesagges > Queue::unpack (
    const std::string & data ) [override], [virtual]
```

Implements [IMesagges](#).

5.38.1.8 unpack() [2/2]

```
std::shared_ptr< IMesagges > Queue::unpack (
    const std::string & data ) [override], [virtual]
```

Implements [IMesagges](#).

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

- bonus/common/messages/Queue.hpp
- common/messages/Queue.hpp
- bonus/common/messages/Queue.cpp
- common/messages/Queue.cpp

5.39 Reception Class Reference

Classes

- class [ErrorReception](#)

Public Member Functions

- **int** **getNbKitchens** () **const**
- **std::vector**< **std::shared_ptr**< **Kitchen** > > **getKitchens** () **const**
- **std::shared_ptr**< **Kitchen** > **getKitchen** (int id) **const**
- **void** **setValues** (int nbCooks, int cookTime, int restockTime, bool debug=false)
- **void** **createKitchen** (int id, int nbCooks, int cookTime, int restockTime)
- **void** **destroyKitchen** (int id)
- **void** **processOrders** (**const** **std::vector**< **std::string** > &orders)
- **void** **orderingLoop** ()
- **bool** **sendOrderToKitchen** (**const** **std::string** &orderData)
- **void** **monitorKitchens** ()
- **void** **updateKitchenStat** (**std::map**< IngridientType, int > ingredients, **std::shared_ptr**< **Kitchen** > kitchens)
- **std::map**< RecipyType, **std::string** > **reloadRecipyTypeNames** ()
- **std::vector**< **std::string** > **checkCommand** (**const** char *command)
- **std::string** **typeToString** (RecipyType type)
- **void** **printMenu** (**std::map**< RecipyType, **std::string** > **RecipyTypeNames**)
- **void** **interMessaege** (**std::shared_ptr**< **Socket** > socket, int id)
- **void** **inactivityMessage** (**std::string** message)
- **void** **orderCompletionMessage** (**std::string** message)
- **void** **refillMessage** (**std::string** message)
- **void** **queueMessage** (**std::string** message)
- **void** **cookStatusMessage** (**std::string** message)
- **int** **getNbKitchens** () **const**
- **std::vector**< **std::shared_ptr**< **Kitchen** > > **getKitchens** () **const**
- **std::shared_ptr**< **Kitchen** > **getKitchen** (int id) **const**
- **void** **setValues** (int nbCooks, int cookTime, int restockTime, bool debug=false)
- **void** **createKitchen** (int id, int nbCooks, int cookTime, int restockTime)
- **void** **destroyKitchen** (int id)
- **void** **killKitchen** ()
- **void** **processOrders** (**const** **std::vector**< **std::string** > &orders)
- **void** **orderingLoop** ()
- **bool** **sendOrderToKitchen** (**std::string** &orderData)
- **void** **monitorKitchens** ()
- **void** **updateKitchenStat** (**std::map**< IngridientType, int > ingredients, **std::shared_ptr**< **Kitchen** > kitchens)
- **std::string** **typeToString** (PizzaType type)
- PizzaType **stringToType** (**const** **std::string** &typeString)
- **std::map**< PizzaType, **std::string** > **reloadRecipyTypeNames** ()
- **void** **printMenu** (**std::map**< PizzaType, **std::string** > **RecipyTypeNames**)
- **void** **interMessaege** (**std::shared_ptr**< **Socket** > socket, int id)
- **std::vector**< **std::string** > **checkCommand** (**const** char *command)
- **void** **inactivityMessage** (**std::string** message)
- **void** **orderCompletionMessage** (**std::string** message)
- **void** **refillMessage** (**std::string** message)
- **void** **queueMessage** (**std::string** message)
- **void** **cookStatusMessage** (**std::string** message)

Private Attributes

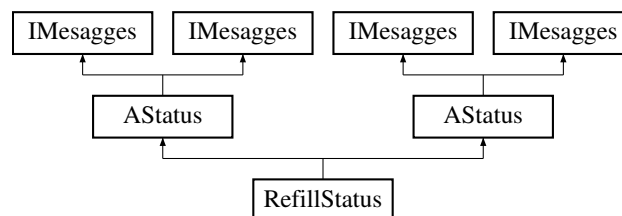
- `int _nbCooks`
- `int _cookTime`
- `int _restockTime`
- `int _nbKitchens`
- `bool _isDebug`
- `std::vector< std::shared_ptr< Kitchen > > _kitchens`
- `std::unordered_map< int, std::shared_ptr< Socket > > _kitchenSockets`
- `std::mutex _kitchensMutex`
- `std::atomic< bool > _isRunning`
- `std::thread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

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

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

5.40 RefillStatus Class Reference

Inheritance diagram for RefillStatus:

**Public Member Functions**

- **RefillStatus** (`int id`, `std::vector< ingStat > status`)
- `MessageType getType () const override`
- `std::shared_ptr< IMesagges > unpack (const std::string &data) override`
- **RefillStatus** (`int id`, `std::vector< ingStat > status`)
- `MessageType getType () const override`
- `std::shared_ptr< IMesagges > unpack (const std::string &data) override`

Public Member Functions inherited from AStatus

- **AStatus** (`int id`, `std::vector< ingStat > status`)
- `std::string pack (const IMesagges &messages) const override`
- `std::string typeToString (MessageType type) const override`
- **AStatus** (`int id`, `std::vector< ingStat > status`)
- `std::string pack (const IMesagges &messages) const override`
- `std::string typeToString (MessageType type) const override`

Additional Inherited Members

Public Attributes inherited from [AStatus](#)

- [int](#) kitchenId
- std::vector< [ingStat](#) > status

5.40.1 Member Function Documentation

5.40.1.1 [getType\(\)](#) [1/2]

```
MessageType RefillStatus::getType ( ) const [override], [virtual]
```

Implements [AStatus](#).

5.40.1.2 [getType\(\)](#) [2/2]

```
MessageType RefillStatus::getType ( ) const [override], [virtual]
```

Implements [AStatus](#).

5.40.1.3 [unpack\(\)](#) [1/2]

```
std::shared_ptr< IMesagges > RefillStatus::unpack (
    const std::string & data ) [override], [virtual]
```

Implements [AStatus](#).

5.40.1.4 [unpack\(\)](#) [2/2]

```
std::shared_ptr< IMesagges > RefillStatus::unpack (
    const std::string & data ) [override], [virtual]
```

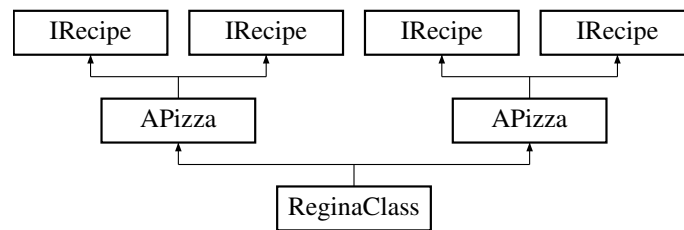
Implements [AStatus](#).

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

- bonus/common/messages/RefillStatus.hpp
- common/messages/RefillStatus.hpp
- bonus/common/messages/RefillStatus.cpp
- common/messages/RefillStatus.cpp

5.41 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 > ingradient](#)) [override](#)
- [void serve](#) () [override](#)
- **ReginaClass** ([int](#) number)
- [void cook](#) ([int](#) cookTime) [override](#)
- [std::shared_ptr< Ingridient > prepare](#) ([int](#) number, [std::shared_ptr< Ingridient > ingradient](#)) [override](#)
- [void serve](#) () [override](#)

Public Member Functions inherited from [APizza](#)

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

5.41.1 Member Function Documentation

5.41.1.1 cook() [1/2]

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

Implements [APizza](#).

5.41.1.2 cook() [2/2]

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

Implements [APizza](#).

5.41.1.3 prepare() [1/2]

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

Implements [APizza](#).

5.41.1.4 prepare() [2/2]

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

Implements [APizza](#).

5.41.1.5 serve() [1/2]

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

Implements [APizza](#).

5.41.1.6 serve() [2/2]

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

Implements [APizza](#).

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

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

5.42 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)
- **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

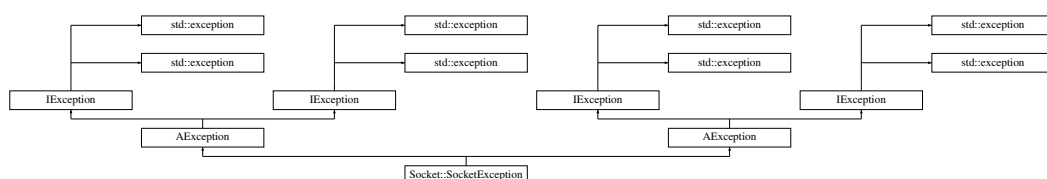
- **int** **_serverFd**
- **int** **_clientFd**
- **struct** **sockaddr_un** **_addr**
- std::string **_sockPath**
- **bool** **_isServer**
- **bool** **_isConnected**

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

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

5.43 Socket::SocketException Class Reference

Inheritance diagram for Socket::SocketException:



Public Member Functions

- **SocketException** ([const](#) std::string &message)
- **SocketException** ([const](#) std::string &message)

Public Member Functions inherited from [AException](#)

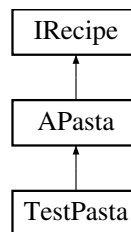
- **AException** ([const](#) std::string &type, [const](#) std::string &message)
- [const](#) char * [what](#) () [const](#) [noexcept](#) [override](#)
- std::string [getType](#) () [const](#) [noexcept](#) [override](#)
- std::string [getMessage](#) () [const](#) [noexcept](#) [override](#)
- std::string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

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

5.44 TestPasta Class Reference

Inheritance diagram for TestPasta:



Public Member Functions

- **TestPasta** ([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](#)

5.44.1 Member Function Documentation

5.44.1.1 cook()

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

Implements [APasta](#).

5.44.1.2 prepare()

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

Implements [APasta](#).

5.44.1.3 serve()

```
void TestPasta::serve ( ) [inline], [override], [virtual]
```

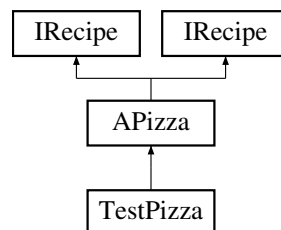
Implements [APasta](#).

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

- tests/Pasta-test.cpp

5.45 TestPizza Class Reference

Inheritance diagram for TestPizza:



Public Member Functions

- **TestPizza** ([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](#)
- [APizza](#) ([int](#) number)
- [int](#) [getNumber](#) () [const override](#)
- [void](#) [setNumber](#) ([int](#) number) [override](#)

5.45.1 Member Function Documentation

5.45.1.1 [cook\(\)](#)

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

Implements [APizza](#).

5.45.1.2 [prepare\(\)](#)

```
std::shared_ptr< Ingridient > TestPizza::prepare (
    int number,
    std::shared_ptr< Ingridient > ingradient ) [inline], [override], [virtual]
```

Implements [APizza](#).

5.45.1.3 [serve\(\)](#)

```
void TestPizza::serve ( ) [inline], [override], [virtual]
```

Implements [APizza](#).

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

- tests/Pizza-test.cpp

5.46 Utils Class Reference

Public Member Functions

- [void](#) [helper](#) ()
- [void](#) [helper](#) ()

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

- bonus/src/Utils/Utils.hpp
- src/Utils/Utils.hpp
- bonus/src/Utils/Utils.cpp
- src/Utils/Utils.cpp

Chapter 6

File Documentation

6.1 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
00014 class APasta : public IRecipe {
00015     public:
00016         APasta(int number);
00017         virtual ~APasta() override = default;
00018         virtual void cook(int cookTime) override = 0;
00019         virtual std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient>
00020 ingridient) override = 0;
00021         virtual void serve() override = 0;
00022
00023         /* Getter */
00024         int getNumber() const override;
00025
00026         /* Setter */
00027         void setNumber(int number) override;
00028     private:
00029         int _size;
00030         int _number;
00031 };
00032
00033 #endif /* !APASTA_HPP_ */
```

6.2 Arrabiata.hpp

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

```

00018         /* Method */
00019         void cook(int cookTime) override;
00020         std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient> ingridient)
00021         override;
00022         void serve() override;
00023     protected:
00024     private:
00025 };
00026
00027 #endif /* !ARRABIATA_HPP_ */

```

6.3 Bolognese.hpp

```

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

```

6.4 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 CarbonaraClass : public APasta {
00014     public:
00015         CarbonaraClass(int number);
00016         ~CarbonaraClass() override;
00017
00018         /* Method */
00019         void cook(int cookTime) override;
00020         std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient> ingridient)
00021         override;
00022         void serve() override;
00023     private:
00024 };
00025
00026 #endif /* !CARBONAR_HPP_ */

```

6.5 Lasagna.hpp

```

00001 /*

```

```

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

```

6.6 Paffo.hpp

```

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

```

6.7 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 PestoClass : public APasta {
00014     public:
00015         PestoClass(int number);
00016         ~PestoClass() override;
00017
00018         /* Method */
00019         void cook(int cookTime) override;

```

```

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

```

6.8 AException.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** AException
00006 */
00007
00008 #ifndef AEXCEPTION_HPP_
00009     #define AEXCEPTION_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     private:
00036         std::string _message;
00037         std::string _type;
00038     };
00039
00040
00041 #endif /* !AEXCEPTION_HPP_ */

```

6.9 AException.hpp

```

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

```

6.10 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
00014 class APizza : public IRecipe {
00015     public:
00016         APizza(int number);
00017         virtual ~APizza() override = default;
00018         virtual void cook(int cookTime) override = 0;
00019         virtual std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient>
00020 ingridient) override = 0;
00021         virtual void serve() override = 0;
00022
00023         /* Getter */
00024         int getNumber() const override;
00025
00026         /* Setter */
00027         void setNumber(int number) override;
00028     private:
00029         int _number;
00030 };
00031
00032 #endif /* !APIZZA_HPP_ */

```

6.11 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_ */

```

6.12 IException.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** IException
00006 */
00007
00008 #include <exception>
00009 #include <string>
00010
00011 #ifndef IEXCEPTION_HPP_
00012     #define IEXCEPTION_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 /* !IEXCEPTION_HPP_ */

```

6.13 IException.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** IException
00006 */
00007
00008 #include <exception>
00009 #include <string>
00010
00011 #ifndef IEXCEPTION_HPP_
00012     #define IEXCEPTION_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 /* !IEXCEPTION_HPP_ */

```

6.14 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      std::shared_ptr<Ingridient> operator=(const std::vector<ingStat> &ingStat);
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> _ingradient;
00093     };
00094
00095 #endif /* !INGREDIENT_HPP_ */

```

6.15 Ingradient.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** Ingradient
00006 */
00007
00008 #include <vector>
00009 #include <string>
00010 #include <memory>
00011 #include <sstream>
00012 #include <map>
00013
00014 #ifndef INGRIDIENT_HPP_
00015 #define INGRIDIENT_HPP_
00016
00017 enum IngradientType
00018 {
00019     DOUGH = 0,
00020     TOMATO = 1,
00021     CHEESE = 2,
00022     HAM = 3,
00023     MUSHROOM = 4,
00024     STEAK = 5,
00025     EGGPLANT = 6,
00026     GOAT_CHEESE = 7,
00027     CHEF_LOVE = 8,
00028     EGG = 9,
00029     BACON = 10,
00030     BASIL = 11,
00031     PEPPER = 12
00032 };
00033
00034 struct ingStat {
00035     IngradientType type;
00036     int quantity;
00037 };
00038
00039 class Ingradient {
00040 public:
00041     Ingradient();
00042     ~Ingradient() = default;
00043     std::vector<ingStat> fridgeStatus();
00044
00045     /* Getter */
00046     int getDough() const;
00047     int getTomato() const;
00048     int getCheese() const;
00049     int getHam() const;
00050     int getMushroom() const;
00051     int getSteak() const;
00052     int getEggplant() const;
00053     int getGoatCheese() const;
00054     int getChefLove() const;
00055     int getEgg() const;
00056     int getBacon() const;
00057     int getBasil() const;
00058     int getPepper() const;
00059
00060     /* Setter */
00061     void setDough(int dough);
00062     void setTomato(int tomato);
00063     void setCheese(int cheese);
00064     void setHam(int ham);
00065     void setMushroom(int mushroom);
00066     void setSteak(int steak);
00067     void setEggplant(int eggplant);
00068     void setGoatCheese(int goatCheese);
00069     void setChefLove(int chefLove);
00070     void setEgg(int egg);
00071     void setBacon(int bacon);
00072     void setBasil(int basil);
00073     void setPepper(int pepper);

```

```

00074
00075     /* Packing/Unpacking methods */
00076     std::string packIngredients() const;
00077     static std::map<IngridientType, int> unpackIngredients(const std::string& packedData);
00078     std::shared_ptr<Ingridient> operator=(const std::vector<ingStat> &ingStat);
00079 private:
00080     int _dough;
00081     int _tomato;
00082     int _cheese;
00083     int _ham;
00084     int _mushroom;
00085     int _steak;
00086     int _eggplant;
00087     int _goatCheese;
00088     int _chefLove;
00089     int _egg;
00090     int _bacon;
00091     int _basil;
00092     int _pepper;
00093     std::vector<ingStat> _ingridient;
00094 };
00095
00096 #endif /* !INGREDIENT_HPP_ */

```

6.16 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 enum RecipyType
00026 {
00027     /* Pizza types*/
00028     Nothing = 0,
00029     Regina = 1,
00030     Margarita = 2,
00031     Americana = 4,
00032     Fantasia = 8,
00033
00034     /* Pasta Types */
00035     Carbonara = 10,
00036     Pesto = 12,
00037     Bolognese = 14,
00038     Arrabiata = 16,
00039     Paffo = 18,
00040     Lasagna = 20
00041 };
00042
00043
00044 class IRecipe {
00045 public:
00046
00047     virtual ~IRecipe() = default;
00048     virtual void cook(int cookTime) = 0;
00049     virtual std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient>
00050 ingridient) = 0;
00051     virtual void serve() = 0;
00052
00053     /* Getter */
00054     virtual int getNumber() const = 0;
00055     /* Setter */
00056     virtual void setNumber(int number) = 0;
00057

```

```

00057 };
00058
00059 #endif /* !IRECEIPY_HPP_ */

```

6.17 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>
00032             ingridient) = 0;
00033         virtual void serve() = 0;
00034
00035         /* Getter */
00036         virtual int getNumber() const = 0;
00037         /* Setter */
00038         virtual void setNumber(int number) = 0;
00039 };
00040
00041 #endif /* !IRECEIPY_HPP_ */

```

6.18 AStatus.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** Status
00006 */
00007
00008
00009 #include <memory>
00010
00011 #include "../Ingridient.hpp"
00012 #include "IMesagges.hpp"
00013
00014 #ifndef ASTATUS_HPP_
00015 #define ASTATUS_HPP_
00016
00017 class AStatus : public IMesagges {
00018     public:
00019         int kitchenId;
00020         std::vector<ingStat> status;
00021
00022         /* Constrcutor */
00023         AStatus(int id, std::vector<ingStat> status);
00024         ~AStatus() override = default;
00025
00026         /* Pack */
00027         std::string pack(const IMesagges &messages) const override;
00028

```

```

00029         /* Unpack */
00030         virtual std::shared_ptr<IMesagges> unpack(const std::string &data) override = 0;
00031
00032         /* Ox to str */
00033         std::string typeToString(MessageType type) const override;
00034
00035         /* Virtual function that needs to be override */
00036         virtual MessageType getType() const override = 0;
00037     protected:
00038     private:
00039 };
00040
00041 #endif /* !ASTATUS_HPP_ */

```

6.19 AStatus.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** Status
00006 */
00007
00008
00009 #include <memory>
00010
00011 #include "../Ingridient.hpp"
00012 #include "IMesagges.hpp"
00013
00014 #ifndef ASTATUS_HPP_
00015 #define ASTATUS_HPP_
00016
00017 class AStatus : public IMesagges {
00018     public:
00019         int kitchenId;
00020         std::vector<ingStat> status;
00021
00022         /* Constrcutor */
00023         AStatus(int id, std::vector<ingStat> status);
00024         ~AStatus() override = default;
00025
00026         /* Pack */
00027         std::string pack(const IMesagges &messages) const override;
00028
00029         /* Unpack */
00030         virtual std::shared_ptr<IMesagges> unpack(const std::string &data) override = 0;
00031
00032         /* Ox to str */
00033         std::string typeToString(MessageType type) const override;
00034
00035         /* Virtual function that needs to be override */
00036         virtual MessageType getType() const override = 0;
00037     protected:
00038     private:
00039 };
00040
00041 #endif /* !ASTATUS_HPP_ */

```

6.20 CookStatus.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** CookStatus
00006 */
00007
00008 #include <vector>
00009 #include <memory>
00010
00011 #include "IMesagges.hpp"
00012
00013 #ifndef COOKSTATUS_HPP_
00014 #define COOKSTATUS_HPP_
00015
00016 struct CookStatusData {
00017     int cookId;
00018     bool isBusy;
00019     bool isRestocking;

```

```

00020 };
00021
00022 class CookStatus : public IMessages {
00023     public:
00024         int _kitchenId;
00025         std::vector<CookStatusData> _cooksStatus;
00026
00027         CookStatus(int kitchenID, const std::vector<CookStatusData> &cooksStatus);
00028         ~CookStatus() override = default;
00029
00030         /* Pack */
00031         std::string pack(const IMessages &messages) const override;
00032
00033         /* Unpack */
00034         std::shared_ptr<IMessages> unpack(const std::string &data) override;
00035
00036         MessageType getType() const override;
00037         std::string typeToString(MessageType type) const override;
00038     protected:
00039     private:
00040 };
00041
00042 #endif /* !COOKSTATUS_HPP_ */

```

6.21 CookStatus.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** CookStatus
00006 */
00007
00008 #include <vector>
00009 #include <memory>
00010
00011 #include "IMessages.hpp"
00012
00013 #ifndef COOKSTATUS_HPP_
00014 #define COOKSTATUS_HPP_
00015
00016 struct CookStatusData {
00017     int cookId;
00018     bool isBusy;
00019     bool isRestocking;
00020 };
00021
00022 class CookStatus : public IMessages {
00023     public:
00024         int _kitchenId;
00025         std::vector<CookStatusData> _cooksStatus;
00026
00027         CookStatus(int kitchenID, const std::vector<CookStatusData> &cooksStatus);
00028         ~CookStatus() override = default;
00029
00030         /* Pack */
00031         std::string pack(const IMessages &messages) const override;
00032
00033         /* Unpack */
00034         std::shared_ptr<IMessages> unpack(const std::string &data) override;
00035
00036         MessageType getType() const override;
00037         std::string typeToString(MessageType type) const override;
00038     protected:
00039     private:
00040 };
00041
00042 #endif /* !COOKSTATUS_HPP_ */

```

6.22 DoneStatus.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** DoneStatus
00006 */
00007
00008 #include "AStatus.hpp"

```



```

00009
00010 #ifndef DONESTATUS_HPP_
00011 #define DONESTATUS_HPP_
00012
00013 class DoneStatus : public AStatus {
00014     public:
00015         DoneStatus(int id, std::vector<ingStat> status);
00016         MessageType getType() const override;
00017         std::shared_ptr<IMesagges> unpack(const std::string &data) override;
00018
00019     protected:
00020     private:
00021 };
00022
00023 #endif /* !DONESTATUS_HPP_ */

```

6.23 DoneStatus.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** DoneStatus
00006 */
00007
00008 #include "AStatus.hpp"
00009
00010 #ifndef DONESTATUS_HPP_
00011 #define DONESTATUS_HPP_
00012
00013 class DoneStatus : public AStatus {
00014     public:
00015         DoneStatus(int id, std::vector<ingStat> status);
00016         MessageType getType() const override;
00017         std::shared_ptr<IMesagges> unpack(const std::string &data) override;
00018
00019     protected:
00020     private:
00021 };
00022
00023 #endif /* !DONESTATUS_HPP_ */

```

6.24 IMesagges.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** IMesagges
00006 */
00007
00008 #include <string>
00009 #include <memory>
00010
00011
00012 #ifndef IMESAGGES_HPP_
00013 #define IMESAGGES_HPP_
00014
00015
00016 enum class MessageType : uint8_t {
00017     Order = 0x01,
00018     Status = 0x02,
00019     Inactivity = 0x03,
00020     Refill = 0x04,
00021     Queue = 0x05,
00022     CookStatus = 0x06,
00023 };
00024
00025
00026 class IMesagges {
00027     public:
00028         virtual ~IMesagges() = default;
00029         virtual std::string pack(const IMesagges &messages) const = 0;
00030         virtual std::shared_ptr<IMesagges> unpack(const std::string &data) = 0;
00031         virtual MessageType getType() const = 0;
00032         virtual std::string typeToString(MessageType type) const = 0;
00033
00034     protected:
00035     private:

```

```

00036 };
00037
00038 #endif /* !IMESAGGES_HPP_ */

```

6.25 IMessages.hpp

```

00001 /*
00002  ** EPITECH PROJECT, 2025
00003  ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004  ** File description:
00005  ** IMessages
00006  */
00007
00008 #include <string>
00009 #include <memory>
00010
00011
00012 #ifndef IMESAGGES_HPP_
00013 #define IMESAGGES_HPP_
00014
00015
00016 enum class MessageType : uint8_t {
00017     Order = 0x01,
00018     Status = 0x02,
00019     Inactivity = 0x03,
00020     Refill = 0x04,
00021     Queue = 0x05,
00022     CookStatus = 0x06,
00023 };
00024
00025
00026 class IMessages {
00027 public:
00028     virtual ~IMessages() = default;
00029     virtual std::string pack(const IMessages &messages) const = 0;
00030     virtual std::shared_ptr<IMessages> unpack(const std::string &data) = 0;
00031     virtual MessageType getType() const = 0;
00032     virtual std::string typeToString(MessageType type) const = 0;
00033
00034 protected:
00035 private:
00036 };
00037
00038 #endif /* !IMESAGGES_HPP_ */

```

6.26 Inactivity.hpp

```

00001 /*
00002  ** EPITECH PROJECT, 2025
00003  ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004  ** File description:
00005  ** Inactivity
00006  */
00007
00008
00009 #include <memory>
00010
00011 #include "IMessages.hpp"
00012
00013 #ifndef INACTIVITY_HPP_
00014 #define INACTIVITY_HPP_
00015
00016 class Inactivity : public IMessages {
00017 public:
00018     int id;
00019
00020     /* Constructor */
00021     Inactivity(int kitchenId);
00022     ~Inactivity() override = default;
00023
00024     /* Pack */
00025     std::string pack(const IMessages &messages) const override;
00026
00027     /* Unpack */
00028     std::shared_ptr<IMessages> unpack(const std::string &data) override;
00029
00030     MessageType getType() const override;
00031     std::string typeToString(MessageType type) const override;
00032 protected:

```

```

00033     private:
00034 };
00035
00036 #endif /* !INACTIVITY_HPP_ */

```

6.27 Inactivity.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** Inactivity
00006 */
00007
00008
00009 #include <memory>
00010
00011 #include "IMesagges.hpp"
00012
00013 #ifndef INACTIVITY_HPP_
00014 #define INACTIVITY_HPP_
00015
00016 class Inactivity : public IMesagges {
00017     public:
00018         int id;
00019
00020         /* Constructor */
00021         Inactivity(int kitchenId);
00022         ~Inactivity() override = default;
00023
00024         /* Pack */
00025         std::string pack(const IMesagges &messages) const override;
00026
00027         /* Unpack */
00028         std::shared_ptr<IMesagges> unpack(const std::string &data) override;
00029
00030         MessageType getType() const override;
00031         std::string typeToString(MessageType type) const override;
00032     protected:
00033     private:
00034 };
00035
00036 #endif /* !INACTIVITY_HPP_ */

```

6.28 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 #include "IMesagges.hpp"
00015
00016
00017 class Order : public IMesagges {
00018     public:
00019         RecipyType type;
00020         Size size;
00021         int number;
00022
00023         /* Constructor */
00024         Order(RecipyType t, Size s, int n);
00025         ~Order() override = default;
00026
00027         /* Pack order Message */
00028         std::string pack(const IMesagges &order) const override;
00029
00030         /* Unpack Order */
00031         std::shared_ptr<IMesagges> unpack(const std::string &data) override;
00032
00033         MessageType getType() const override;
00034         std::string typeToString(MessageType type) const override;
00035 };

```

6.29 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 #include "IMesagges.hpp"
00015
00016
00017 class Order : public IMesagges {
00018     public:
00019         PizzaType type;
00020         Size size;
00021         int number;
00022
00023         /* Constructor */
00024         Order(PizzaType t, Size s, int n);
00025         ~Order() override = default;
00026
00027         /* Pack order Message */
00028         std::string pack(const IMesagges &order) const override;
00029
00030         /* Unpack Order */
00031         std::shared_ptr<IMesagges> unpack(const std::string &data) override;
00032
00033         MessageType getType() const override;
00034         std::string typeToString(MessageType type) const override;
00035 };

```

6.30 Queue.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** Queue
00006 */
00007
00008 #include <memory>
00009
00010 #include "IMesagges.hpp"
00011
00012 #ifndef QUEUE_HPP_
00013 #define QUEUE_HPP_
00014
00015 class Queue : public IMesagges {
00016     public:
00017         int kitchenId;
00018         int nbCurrentOrders;
00019
00020         /* Constructor */
00021         Queue(int id, int currentOrders);
00022         ~Queue() override = default;
00023
00024         /* Pack */
00025         std::string pack(const IMesagges &messages) const override;
00026
00027         /* Unpack */
00028         std::shared_ptr<IMesagges> unpack(const std::string &data) override;
00029
00030         MessageType getType() const override;
00031         std::string typeToString(MessageType type) const override;
00032     protected:
00033     private:
00034 };
00035
00036 #endif /* !QUEUE_HPP_ */

```

6.31 Queue.hpp

```

00001 /*

```

```

00002  ** EPITECH PROJECT, 2025
00003  ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004  ** File description:
00005  ** Queue
00006  */
00007
00008 #include <memory>
00009
00010 #include "IMesagges.hpp"
00011
00012 #ifndef QUEUE_HPP_
00013 #define QUEUE_HPP_
00014
00015 class Queue : public IMesagges {
00016     public:
00017         int kitchenId;
00018         int nbCurrentOrders;
00019
00020         /* Constructor */
00021         Queue(int id, int currentOrders);
00022         ~Queue() override = default;
00023
00024         /* Pack */
00025         std::string pack(const IMesagges &messages) const override;
00026
00027         /* Unpack */
00028         std::shared_ptr<IMesagges> unpack(const std::string &data) override;
00029
00030         MessageType getType() const override;
00031         std::string typeToString(MessageType type) const override;
00032     protected:
00033     private:
00034 };
00035
00036 #endif /* !QUEUE_HPP_ */

```

6.32 RefillStatus.hpp

```

00001 /*
00002  ** EPITECH PROJECT, 2025
00003  ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004  ** File description:
00005  ** Refill
00006  */
00007
00008 #include "AStatus.hpp"
00009
00010 #ifndef REFILLSTATUS_HPP_
00011 #define REFILLSTATUS_HPP_
00012
00013 class RefillStatus : public AStatus {
00014     public:
00015         RefillStatus(int id, std::vector<ingStat> status);
00016         MessageType getType() const override;
00017         std::shared_ptr<IMesagges> unpack(const std::string &data) override;
00018
00019     protected:
00020     private:
00021 };
00022
00023 #endif /* !REFILLSTATUS_HPP_ */

```

6.33 RefillStatus.hpp

```

00001 /*
00002  ** EPITECH PROJECT, 2025
00003  ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004  ** File description:
00005  ** Refill
00006  */
00007
00008 #include "AStatus.hpp"
00009
00010 #ifndef REFILLSTATUS_HPP_
00011 #define REFILLSTATUS_HPP_
00012
00013 class RefillStatus : public AStatus {
00014     public:
00015         RefillStatus(int id, std::vector<ingStat> status);

```

```

00016         MessageType getType() const override;
00017         std::shared_ptr<IMesagges> unpack(const std::string &data) override;
00018
00019     protected:
00020     private:
00021 };
00022
00023 #endif /* !REFILLSTATUS_HPP_ */

```

6.34 ForkProcess.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** ForkProcess
00006 */
00007
00008 #include <unistd.h>
00009 #include <sys/types.h>
00010 #include <sys/wait.h>
00011 #include <stdexcept>
00012 #include <functional>
00013 #include <iostream>
00014
00015 #include "IProcess.hpp"
00016
00017 #ifndef FORKPROCESS_HPP_
00018 #define FORKPROCESS_HPP_
00019
00020 class ForkProcess : public IProcess {
00021     class ProcessException : public std::runtime_error {
00022     public:
00023         explicit ProcessException(const std::string& msg) : std::runtime_error(msg) {}
00024     };
00025     public:
00026         ForkProcess();
00027         ~ForkProcess() override;
00028
00029         /* Override Methods */
00030         pid_t create(const std::function<void()> &childLogic) override;
00031         int wait() override;
00032         void close() override;
00033         pid_t getPid() const override;
00034
00035     protected:
00036     private:
00037         pid_t _pid;
00038 };
00039
00040 #endif /* !FORKPROCESS_HPP_ */

```

6.35 ForkProcess.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** ForkProcess
00006 */
00007
00008 #include <unistd.h>
00009 #include <sys/types.h>
00010 #include <sys/wait.h>
00011 #include <stdexcept>
00012 #include <functional>
00013 #include <iostream>
00014
00015 #include "IProcess.hpp"
00016
00017 #ifndef FORKPROCESS_HPP_
00018 #define FORKPROCESS_HPP_
00019
00020 class ForkProcess : public IProcess {
00021     class ProcessException : public std::runtime_error {
00022     public:
00023         explicit ProcessException(const std::string& msg) : std::runtime_error(msg) {}
00024     };
00025     public:

```

```

00026         ForkProcess();
00027         ~ForkProcess() override;
00028
00029         /* Override Methods */
00030         pid_t create(const std::function<void()> &childLogic) override;
00031         int wait() override;
00032         void close() override;
00033         pid_t getPid() const override;
00034
00035     protected:
00036     private:
00037         pid_t _pid;
00038 };
00039
00040 #endif /* !FORKPROCESS_HPP_ */

```

6.36 IProcess.hpp

```

00001 /*
00002  ** EPITECH PROJECT, 2025
00003  ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004  ** File description:
00005  ** IProcess
00006  */
00007
00008 #include <sys/types.h>
00009 #include <sys/wait.h>
00010
00011 #include <iostream>
00012 #include <functional>
00013
00014 #ifndef IPROCESS_HPP_
00015 #define IPROCESS_HPP_
00016
00017 class IProcess {
00018     public:
00019         virtual pid_t create(const std::function<void()> &childLogic) = 0;
00020         virtual int wait() = 0;
00021         virtual void close() = 0;
00022         virtual pid_t getPid() const = 0;
00023         virtual ~IProcess() = default;
00024     protected:
00025     private:
00026 };
00027
00028 #endif /* !IPROCESS_HPP_ */

```

6.37 IProcess.hpp

```

00001 /*
00002  ** EPITECH PROJECT, 2025
00003  ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004  ** File description:
00005  ** IProcess
00006  */
00007
00008 #include <sys/types.h>
00009 #include <sys/wait.h>
00010
00011 #include <iostream>
00012 #include <functional>
00013
00014 #ifndef IPROCESS_HPP_
00015 #define IPROCESS_HPP_
00016
00017 class IProcess {
00018     public:
00019         virtual pid_t create(const std::function<void()> &childLogic) = 0;
00020         virtual int wait() = 0;
00021         virtual void close() = 0;
00022         virtual pid_t getPid() const = 0;
00023         virtual ~IProcess() = default;
00024     protected:
00025     private:
00026 };
00027
00028 #endif /* !IPROCESS_HPP_ */

```

6.38 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_ */

```

6.39 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_ */

```

6.40 DLoader.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** arcade
00004 ** File description:
00005 ** DLoader
00006 */
00007
00008 #ifndef DLOADER_HPP_
00009 #define DLOADER_HPP_
00010
00011 #include <dlfcn.h>
00012 #include <iostream>
00013 #include <ostream>
00014 #include "ILoader.hpp"
00015
00016 template <typename T>
00017
00018 class DLoader : public ILoader {
00019     private:
00020         void *_handler = nullptr;
00021
00022     public:
00023         ~DLoader() = default;
00024
00025         void *getHandler() const override {
00026             return _handler;
00027         };
00028         void *Open(const char *path, int flag) override {
00029             _handler = dlopen(path, flag);
00030             return _handler;
00031         };
00032         void *Symbol(const char *symbolName) override {
00033             void *symbol = dlsym(_handler, symbolName);
00034             const char *error = dlerror();
00035             if (error) {
00036                 std::cerr << "dlerror: " << error << std::endl;
00037                 return nullptr;
00038             }
00039             return symbol;
00040         };
00041         T getSymbol(const char *symbolName) {
00042             return reinterpret_cast<T>(dlsym(_handler, symbolName));
00043         };
00044         int Close() override {
00045             if (_handler == nullptr)
00046                 return -1;

```

```

00047         return dlclose(_handler);
00048     };
00049     const char *Error() override {
00050         return dlerror();
00051     };
00052 };
00053
00054 #endif /* !DLLOADER_HPP_ */

```

6.41 DLLoader.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** arcade
00004 ** File description:
00005 ** DLLoader
00006 */
00007
00008 #ifndef DLLOADER_HPP_
00009 #define DLLOADER_HPP_
00010
00011 #include <dlfcn.h>
00012 #include <iostream>
00013 #include <ostream>
00014 #include "ILoader.hpp"
00015
00016 template <typename T>
00017 class DLLoader : public ILoader {
00018     private:
00019         void *_handler = nullptr;
00020
00021     public:
00022         ~DLLoader() = default;
00023
00024         void *getHandler() const override {
00025             return _handler;
00026         };
00027
00028         void *Open(const char *path, int flag) override {
00029             _handler = dlopen(path, flag);
00030             return _handler;
00031         };
00032
00033         void *Symbol(const char *symbolName) override {
00034             void *symbol = dlsym(_handler, symbolName);
00035             const char *error = dlerror();
00036             if (error) {
00037                 std::cerr << "dlerror: " << error << std::endl;
00038                 return nullptr;
00039             }
00040             return symbol;
00041         };
00042
00043         T getSymbol(const char *symbolName) {
00044             return reinterpret_cast<T>(dlsym(_handler, symbolName));
00045         };
00046
00047         int Close() override {
00048             if (_handler == nullptr)
00049                 return -1;
00050             return dlclose(_handler);
00051         };
00052
00053         const char *Error() override {
00054             return dlerror();
00055         };
00056 };
00057
00058 #endif /* !DLLOADER_HPP_ */

```

6.42 ILoader.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004 ** File description:
00005 ** ILoader
00006 */
00007
00008 #ifndef ILoader_HPP_
00009 #define ILoader_HPP_
00010
00011

```

```

00012 class ILoader {
00013     public:
00014         ~ILoader() = default;
00015
00016         virtual void *Open(const char *path, int flag) = 0;
00017         virtual void *Symbol(const char *symbolName) = 0;
00018         virtual int Close() = 0;
00019         virtual const char *Error() = 0;
00020         virtual void *getHandler() const = 0;
00021
00022     protected:
00023     private:
00024 };
00025
00026 #endif /* !ILoader_HPP_ */

```

6.43 ILoader.hpp

```

00001 /*
00002  ** EPITECH PROJECT, 2025
00003  ** B-OOP-400-NAN-4-1-raytracer-albane.merian
00004  ** File description:
00005  ** ILoader
00006  */
00007
00008 #ifndef ILoader_HPP_
00009 #define ILoader_HPP_
00010
00011
00012 class ILoader {
00013     public:
00014         ~ILoader() = default;
00015
00016         virtual void *Open(const char *path, int flag) = 0;
00017         virtual void *Symbol(const char *symbolName) = 0;
00018         virtual int Close() = 0;
00019         virtual const char *Error() = 0;
00020         virtual void *getHandler() const = 0;
00021
00022     protected:
00023     private:
00024 };
00025
00026 #endif /* !ILoader_HPP_ */

```

6.44 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 #include "../common/IRecipe.hpp"
00016 #include "../lib/DLLoader.hpp"
00017
00018 #ifndef COOKS_HPP_
00019 #define COOKS_HPP_
00020
00021 class Cooks {
00022
00023     class ErrorCooks : public AException {
00024     public:
00025         ErrorCooks(const std::string &message);
00026     };
00027
00028     public:
00029         Cooks(std::shared_ptr<Ingridient> ingradient, int id,
00030             int cookTime, int restockTime);
00031         ~Cooks() = default;
00032

```

```

00033         /* Method */
00034         std::shared_ptr<Ingridient> startOrder(std::shared_ptr<Ingridient> ingridient,
std::vector<std::string> order);
00035         bool hasEnoughIngredients(const std::string &orderData, std::shared_ptr<Ingridient>
ingridient);
00036         void waitForTheOven(Size size);
00037         std::shared_ptr<IRecipe> loadPlugin(const std::string &path, int number);
00038         std::string getType(const std::string& path, DLLoader<IRecipe> loader);
00039         /* Getter */
00040         int getID() const;
00041         bool isBusy() const;
00042         bool isRestocking() const;
00043
00044         /* Setter */
00045         void setIsBusy(bool isBusy);
00046         /* Loader Pluggins */
00047         std::string toString(RecipyType type);
00048         std::shared_ptr<IRecipe> findAndLoadPlugin(const std::string &pizzaType, int number);
00049     private:
00050         int _ID;
00051         int _cookTime;
00052         int _restockTime;
00053         std::shared_ptr<Ingridient> _ingridient;
00054         std::mutex _statusMutex;
00055         bool _isBusy;
00056         bool _isRestocking;
00057 };
00058
00059 #endif /* !COOKS_HPP_ */

```

6.45 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 #include "../common/IRecipe.hpp"
00016 #include "../lib/DLLoader.hpp"
00017
00018 #include "../common/APizza.hpp"
00019
00020 #ifndef COOKS_HPP_
00021     #define COOKS_HPP_
00022
00023     class Cooks {
00024     public:
00025         class ErrorCooks : public AException {
00026         public:
00027             ErrorCooks(const std::string &message);
00028         };
00029
00030
00031         Cooks(std::shared_ptr<Ingridient> ingridient, int id,
00032             int cookTime, int restockTime);
00033         ~Cooks() = default;
00034
00035         /* Method */
00036         std::shared_ptr<Ingridient> startOrder(std::shared_ptr<Ingridient> ingridient,
std::vector<std::string> order);
00037         bool hasEnoughIngredients(const std::string &orderData, std::shared_ptr<Ingridient>
ingridient);
00038
00039         std::shared_ptr<IRecipe> loadPlugin(const std::string &path, int number);
00040         std::string getType(const std::string& path, DLLoader<IRecipe> loader);
00041         /* Getter */
00042         int getID() const;
00043         bool isBusy() const;
00044         bool isRestocking() const;
00045
00046         /* Setter */
00047         void setIsBusy(bool isBusy);
00048         /* Loader Pluggins */
00049         std::string toString(PizzaType type);

```

```

00050         std::shared_ptr<IRecipe> findAndLoadPlugin(const std::string &pizzaType, int number);
00051     private:
00052         int _ID;
00053         int _cookTime;
00054         int _restockTime;
00055         std::shared_ptr<Ingridient> _ingradient;
00056         std::mutex _statusMutex;
00057         bool _isBusy;
00058         bool _isRestocking;
00059 };
00060
00061 #endif /* !COOKS_HPP_ */

```

6.46 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 <queue>
00012 #include <mutex>
00013 #include <vector>
00014 #include <thread>
00015 #include <condition_variable>
00016 #include <chrono>
00017
00018 #include "../cooks/Cooks.hpp"
00019 #include "../../common/processes/IProcess.hpp"
00020 #include "../../common/Socket.hpp"
00021
00022 #include "../../common/AException.hpp"
00023
00024 class Kitchen {
00025
00026     class ErrorKitchen : public AException {
00027     public:
00028         ErrorKitchen(const std::string &message);
00029     };
00030
00031     public:
00032         Kitchen(int id, int nbCooks, int cookTime, int restockTime, bool debug);
00033         ~Kitchen();
00034         void restock();
00035         void startKitchenProcess();
00036         void startKitchen();
00037         void run();
00038         void processOrder(const std::string &orderData);
00039         bool canAcceptOrder(int numPizzas);
00040         void stopKitchen();
00041         void sendOrder();
00042         void createCooks();
00043         void setIsFull(bool isFull);
00044
00045         // Process management
00046         void setProcess(std::shared_ptr<IProcess> process) {
00047             _process = process;
00048         }
00049
00050         /* Setter */
00051         void setCurrentOrders(int currentOrders);
00052         void incrementCurrentOrders(int amount);
00053         /* Getter */
00054         int getID() const;
00055         int getNbCooks() const;
00056         int getCookTime() const;
00057         int getRestockTime() const;
00058         int getMaxCmd() const;
00059         std::shared_ptr<Ingridient> getIngridient() const;
00060         std::vector<std::shared_ptr<Cooks>> getCooks() const;
00061         int getCurrentOrders() const;
00062         bool isFull() const;
00063
00064         /* Send Messages */
00065         void sendQueueStatMessage();
00066         void sendDoneMessage();
00067         void sendRefillMessage();
00068         void sendInactive();

```

```

00069         void sendCookStatus();
00070     protected:
00071     private:
00072         int _ID;
00073         int _nbCooks;
00074         int _cookTime;
00075         int _restockTime;
00076         int _maxCmd;
00077         int _currentOrders;
00078         pid_t _pid; // Add this line to track the process ID
00079         std::shared_ptr<Ingridient> _ingridient;
00080         std::vector<std::shared_ptr<Cooks>> _cooks;
00081         std::queue<std::string> _orderQueue;
00082         std::mutex _orderMutex;
00083         std::mutex _ingMutex;
00084         std::condition_variable _cookCV;
00085         std::vector<std::thread> _cookThreads;
00086         std::thread _restockThread;
00087         Socket _socket;
00088         std::chrono::steady_clock::time_point _lastActivity;
00089         std::shared_ptr<IProcess> _process;
00090         bool _isRunning;
00091         bool _isDebug;
00092         bool _isFull;
00093     };
00094
00095     std::ostream& operator<<(std::ostream& os, const Kitchen& kitchen);
00096
00097 #endif /* !KITCHEN_HPP_ */

```

6.47 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 <queue>
00012 #include <mutex>
00013 #include <vector>
00014 #include <thread>
00015 #include <condition_variable>
00016 #include <chrono>
00017
00018 #include "../cooks/Cooks.hpp"
00019 #include "../../common/processes/IProcess.hpp"
00020 #include "../../common/Socket.hpp"
00021
00022 #include "../../common/AException.hpp"
00023
00024 class Kitchen {
00025 public:
00026     class ErrorKitchen : public AException {
00027     public:
00028         ErrorKitchen(const std::string &message);
00029     };
00030
00031     Kitchen(int id, int nbCooks, int cookTime, int restockTime, bool debug);
00032     ~Kitchen();
00033     void restock();
00034     void startKitchenProcess();
00035     void startKitchen();
00036     void run();
00037     void processOrder(const std::string &orderData);
00038     bool canAcceptOrder(int numPizzas);
00039     void stopKitchen();
00040     void sendOrder();
00041     void createCooks();
00042     void setIsFull(bool isFull);
00043
00044     // Process management
00045     void setProcess(std::shared_ptr<IProcess> process) {
00046         _process = process;
00047     }
00048
00049     /* Setter */
00050     void setCurrentOrders(int currentOrders);

```

```

00052         void incrementCurrentOrders(int amount);
00053         /* Getter */
00054         int getID() const;
00055         int getNbCooks() const;
00056         int getCookTime() const;
00057         int getRestockTime() const;
00058         int getMaxCmd() const;
00059         std::shared_ptr<Ingridient> getIngridient() const;
00060         std::vector<std::shared_ptr<Cooks>> getCooks() const;
00061         int getCurrentOrders() const;
00062         bool isFull() const;
00063
00064         /* Send Messages */
00065         void sendQueueStatMessage();
00066         void sendDoneMessage();
00067         void sendRefillMessage();
00068         void sendInactive();
00069         void sendCookStatus();
00070     protected:
00071     private:
00072         int _ID;
00073         int _nbCooks;
00074         int _cookTime;
00075         int _restockTime;
00076         int _maxCmd;
00077         int _currentOrders;
00078         pid_t _pid; // Add this line to track the process ID
00079         std::shared_ptr<Ingridient> _ingridient;
00080         std::vector<std::shared_ptr<Cooks>> _cooks;
00081         std::queue<std::string> _orderQueue;
00082         std::mutex _orderMutex;
00083         std::mutex _ingMutex;
00084         std::condition_variable _cookCV;
00085         std::vector<std::thread> _cookThreads;
00086         std::thread _restockThread;
00087         Socket _socket;
00088         std::chrono::steady_clock::time_point _lastActivity;
00089         std::shared_ptr<IProcess> _process;
00090         bool _isRunning;
00091         bool _isDebug;
00092         bool _isFull;
00093     };
00094
00095     std::ostream& operator<<(std::ostream& os, const Kitchen& kitchen);
00096
00097 #endif /* !KITCHEN_HPP_ */

```

6.48 Plaza.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** Plaza
00006 */
00007
00008 #ifndef PLAZZA_HPP_
00009 #define PLAZZA_HPP_
00010
00011 #include "reception/Reception.hpp"
00012 #include "../common/AException.hpp"
00013
00014 class Plaza {
00015
00016     class ErrorParsing : public AException {
00017     public:
00018         ErrorParsing(const std::string &message);
00019     };
00020
00021     public:
00022         Plaza();
00023         ~Plaza();
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_ */

```

6.49 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_ */

```

6.50 Reception.hpp

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004 ** File description:
00005 ** Reception
00006 */
00007
00008 #ifndef RECEPTION_HPP_
00009 #define RECEPTION_HPP_
00010
00011 #include <thread>
00012 #include <atomic>
00013 #include <mutex>
00014 #include <unordered_map>
00015
00016
00017 #include "../common/AException.hpp"
00018 #include "../common/APizza.hpp"
00019 #include "../common/Socket.hpp"
00020 #include "../kitchen/Kitchen.hpp"
00021
00022
00023 class Reception {
00024
00025     class ErrorReception : public AException {
00026     public:
00027         ErrorReception(const std::string &message);
00028     };
00029
00030     public:
00031         Reception();
00032         ~Reception();
00033
00034         /* Getter */
00035         int getNbKitchens() const;

```



```

00036         std::vector<std::shared_ptr<Kitchen>> getKitchens() const;
00037         std::shared_ptr<Kitchen> getKitchen(int id) const;
00038
00039         /* Setter */
00040         void setValues(int nbCooks, int cookTime, int restockTime,
00041             bool debug = false);
00042
00043         /* Methods */
00044         void createKitchen(int id, int nbCooks, int cookTime, int restockTime);
00045         void destroyKitchen(int id);
00046         void processOrders(const std::vector<std::string> &orders);
00047         void orderingLoop();
00048         bool sendOrderToKitchen(const std::string &orderData);
00049         void monitorKitchens();
00050         void updateKitchenStat(std::map<IngridientType, int> ingredients,
00051             std::shared_ptr<Kitchen> kitchens);
00052
00053         /* Parser Methods */
00054         std::map<RecipyType, std::string> reloadRecipyTypeNames();
00055         std::vector<std::string> checkCommand(const char *command);
00056         std::string typeToString(RecipyType type);
00057         void printMenu(std::map<RecipyType, std::string> RecipyTypeNames);
00058
00059         /* Messefe func handler */
00060         void interMessaage(std::shared_ptr<Socket> socket, int id);
00061         void inactivityMessage(std::string message);
00062         void orderCompletionMessage(std::string message);
00063         void refillMessage(std::string message);
00064         void queueMessage(std::string message);
00065         void cookStatusMessage(std::string message);
00066     protected:
00067     private:
00068         int _nbCooks;
00069         int _cookTime;
00070         int _restockTime;
00071         int _nbKitchens;
00072         bool _isDebug;
00073         std::vector<std::shared_ptr<Kitchen>> _kitchens;
00074         std::unordered_map<int, std::shared_ptr<Socket>> _kitchenSockets;
00075         std::mutex _kitchensMutex;
00076         std::atomic<bool> _isRunning;
00077         std::thread _monitorThread;
00078 };
00079
00080 #endif /* !RECEPTION_HPP_ */

```

6.51 Reception.hpp

```

00001 /*
00002  ** EPITECH PROJECT, 2025
00003  ** B-CCP-400-NAN-4-1-theplazza-albane.merian
00004  ** File description:
00005  ** Reception
00006  */
00007
00008 #ifndef RECEPTION_HPP_
00009 #define RECEPTION_HPP_
00010
00011 #include <thread>
00012 #include <atomic>
00013 #include <mutex>
00014 #include <unordered_map>
00015
00016
00017 #include "../common/AException.hpp"
00018 #include "../common/APizza.hpp"
00019 #include "../common/Socket.hpp"
00020 #include "../kitchen/Kitchen.hpp"
00021
00022
00023 class Reception {
00024     public:
00025
00026         class ErrorReception : public AException {
00027             public:
00028                 ErrorReception(const std::string &message);
00029         };
00030
00031
00032         Reception();
00033         ~Reception();
00034
00035         /* Getter */

```

```

00036         int getNbKitchens() const;
00037         std::vector<std::shared_ptr<Kitchen>> getKitchens() const;
00038         std::shared_ptr<Kitchen> getKitchen(int id) const;
00039
00040         /* Setter */
00041         void setValues(int nbCooks, int cookTime, int restockTime,
00042             bool debug = false);
00043
00044         /* Methods */
00045         void createKitchen(int id, int nbCooks, int cookTime, int restockTime);
00046         void destroyKitchen(int id);
00047         void killKitchen();
00048         void processOrders(const std::vector<std::string> &orders);
00049         void orderingLoop();
00050         bool sendOrderToKitchen(std::string &orderData);
00051         void monitorKitchens();
00052         void updateKitchenStat(std::map<IngridientType, int> ingredients,
00053             std::shared_ptr<Kitchen> kitchens);
00054
00055         /* Reload elem */
00056         std::string typeToString(PizzaType type);
00057         PizzaType stringToType(const std::string& typeString);
00058         std::map<PizzaType, std::string> reloadRecipyTypeNames();
00059         void printMenu(std::map<PizzaType, std::string> RecipyTypeNames);
00060
00061         /* Messefe func handler */
00062         void interMessaege(std::shared_ptr<Socket> socket, int id);
00063         std::vector<std::string> checkCommand(const char *command);
00064         void inactivityMessage(std::string message);
00065         void orderCompletionMessage(std::string message);
00066         void refillMessage(std::string message);
00067         void queueMessage(std::string message);
00068         void cookStatusMessage(std::string message);
00069
00070     protected:
00071     private:
00072         int _nbCooks;
00073         int _cookTime;
00074         int _restockTime;
00075         int _nbKitchens;
00076         bool _isDebug;
00077         std::vector<std::shared_ptr<Kitchen>> _kitchens;
00078         std::unordered_map<int, std::shared_ptr<Socket>> _kitchenSockets;
00079         std::mutex _kitchensMutex;
00080         std::atomic<bool> _isRunning;
00081         std::thread _monitorThread;
00082         DLloader<std::shared_ptr<IProcess>>(*)()> loader;
00083     };
00084
00085 #endif /* !RECEPTION_HPP_ */

```

6.52 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)
00022         override;
00023         void serve() override;
00024
00025     protected:
00026     private:
00027 };
00028 #endif /* !AMERICANA_HPP_ */

```

6.53 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)
00022         override;
00023         void serve() override;
00024
00025     protected:
00026     private:
00027 };
00028 #endif /* !AMERICANA_HPP_ */

```

6.54 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)
00021         override;
00022         void serve() override;
00023
00024     protected:
00025     private:
00026 };
00027 #endif /* !FANTASIA_HPP_ */

```

6.55 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)
00021         override;
00022         void serve() override;
00023
00024     protected:
00025     private:
00026 };
00027 #endif /* !FANTASIA_HPP_ */

```

6.56 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)
00020         override;
00021         void serve() override;
00022
00023     protected:
00024     private:
00025 };
00026
00027 #endif /* !MARGARITACLASS_HPP_ */

```

6.57 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)
00020         override;
00021         void serve() override;
00022
00023     protected:
00024     private:
00025 };
00026
00027 #endif /* !MARGARITACLASS_HPP_ */

```

6.58 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)
00021         override;
00022         void serve() override;
00023     protected:
00024     private:
00025 };
00026 #endif /* !REGINACLASS_HPP_ */

```

6.59 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)
00021         override;
00022         void serve() override;
00023     protected:
00024     private:
00025 };
00026 #endif /* !REGINACLASS_HPP_ */

```

6.60 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_ */
```

6.61 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_ */
```