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 / mult-threading execution.
A recpetion will be defined wich works like a shell
in which we can order pizzas, to cook thos pizza
kitchen will need to be created, each kitchen will habve a
pre-defined number of cooks that can have two pizzas in there
queues.
```

Libraries

```
Only authorized:
    Standard C++ library
```

1.1 Bonus :

```
- Interface implementation of recipy(Pasta)
- Dynamic reload of parsing / recipy
- 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 ablige to a specified coding styke,
to check if the code is complient with the norm execut
the make coding command or the ./styleChecker.sh.
To understand the errors and how to fix them please
refers 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 approriate for the current modification. [Element / Module] = The elemenet you applied the modification. [MESSAGE] = A detail message of what you did.

Gitmojies:

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

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

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

...
```

1.5.1 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 ficher add par erreur qui est pas encore push:
```
git restore --staged <file>
```
- Un-add un fichier qui a été commit :
```
git reset --soft HEAD~1
git restore --staged fichier-a-retirer.txt
git commit -m "Nouveau message de commit (sans le fichier)"
```

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

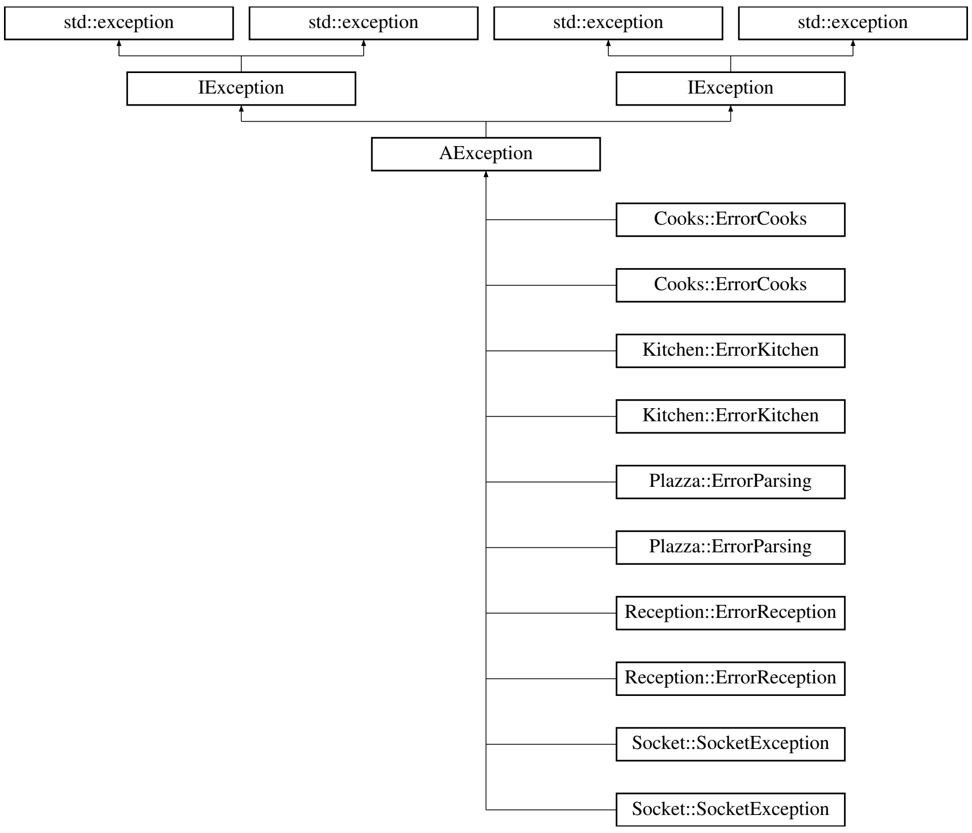
Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 AException Class Reference

Inheritance diagram for AException:



Public Member Functions

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

- std::string getFormattedMessage () const noexcept override
- **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]
```

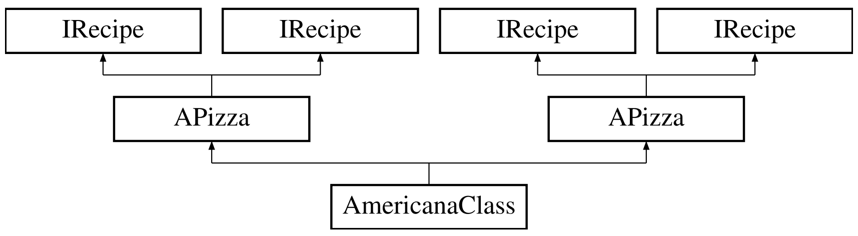
Implements IException.

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]
```

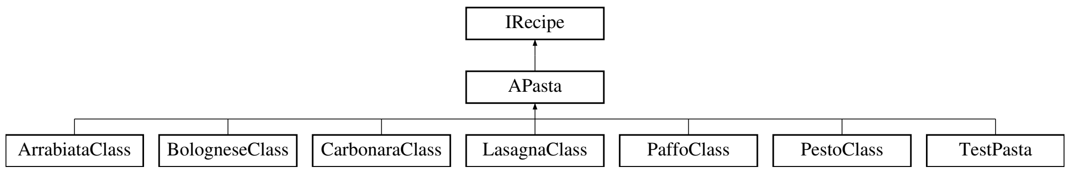
Implements APizza.

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 > ingridient) override=0
- virtual void serve () override=0
- int getNumber () const override
- void setNumber (int number) override

Private Attributes

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

5.3.1 Member Function Documentation

5.3.1.1 cook()

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

Implements IRecipe.

5.3.1.2 getNumber()

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

Implements IRecipe.

5.3.1.3 prepare()

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

Implements IRecipe.

5.3.1.4 serve()

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

Implements IRecipe.

5.3.1.5 setNumber()

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

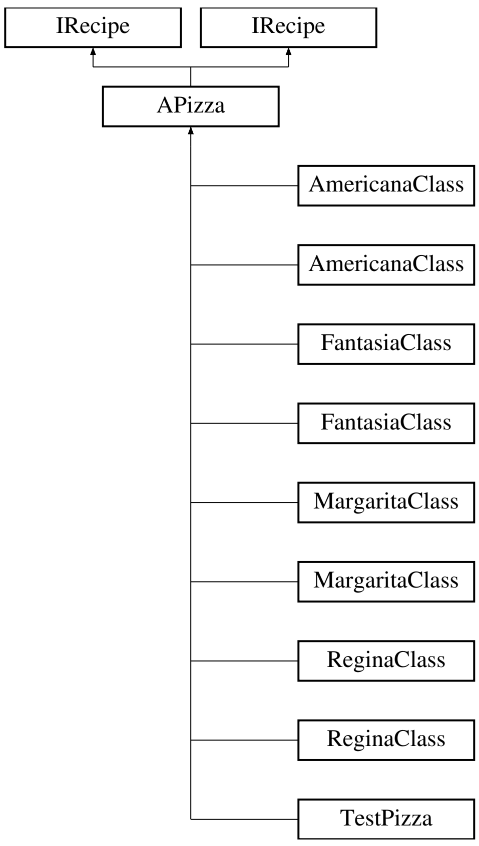
Implements IRecipe.

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

- 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 > ingridient) 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 > ingridient) override=0
- virtual void serve () override=0
- int getNumber () const override
- void setNumber (int number) override

Private Attributes

- int **_number**

5.4.1 Member Function Documentation

5.4.1.1 cook() [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 > ingridient )  [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 > ingridient )  [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]
```

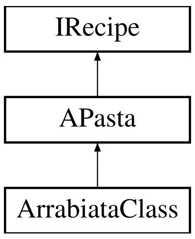
Implements IRecipe.

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]
```

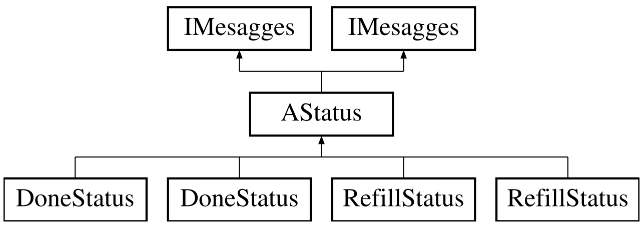
Implements APasta.

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 IMesagges.

5.6.1.2 getType() [2/2]

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

Implements IMesagges.

5.6.1.3 pack() [1/2]

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

Implements IMesagges.

5.6.1.4 pack() [2/2]

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

Implements IMesagges.

5.6.1.5 typeToString() [1/2]

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

Implements IMesagges.

5.6.1.6 typeToString() [2/2]

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

Implements IMesagges.

5.6.1.7 unpack() [1/2]

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

Implements IMesagges.

5.6.1.8 unpack() [2/2]

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

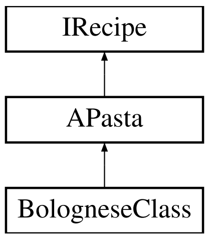
Implements IMesagges.

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]
```

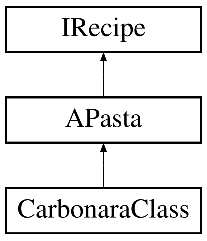
Implements APasta.

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 > ingridient )  [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 > ingridient, int id, int cookTime, int restockTime)
- std::shared_ptr< Ingridient > **startOrder** (std::shared_ptr< Ingridient > ingridient, std::vector< std::string > order)
- bool **hasEnoughIngredients** (const std::string &orderData, std::shared_ptr< Ingridient > ingridient)
- 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 > ingridient, int id, int cookTime, int restockTime)
- std::shared_ptr< Ingridient > **startOrder** (std::shared_ptr< Ingridient > ingridient, std::vector< std::string > order)
- bool **hasEnoughIngredients** (const std::string &orderData, std::shared_ptr< Ingridient > ingridient)
- 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

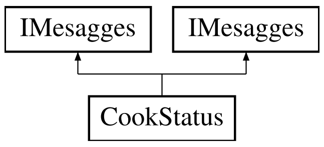
- int **_ID**
- int **_cookTime**
- int **_restockTime**
- std::shared_ptr< Ingridient > **_ingridient**
- std::mutex **_statusMutex**
- bool **_isBusy**
- bool **_isRestocking**

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 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
- **CookStatus** (int kitchenID, const std::vector< CookStatusData > &cooksStatus)
- 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 **_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 IMesagges.

5.10.1.2 getType() [2/2]

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

Implements IMesagges.

5.10.1.3 pack() [1/2]

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

Implements IMesagges.

5.10.1.4 pack() [2/2]

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

Implements IMesagges.

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

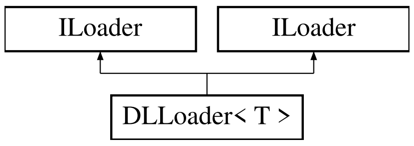
- int **cookId**
- bool **isBusy**
- bool **isRestocking**

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 * DLLoader< T >::Symbol (
            const char * symbolName )  [inline], [override], [virtual]
```

Implements ILoader.

5.12.1.10 Symbol() [2/2]

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

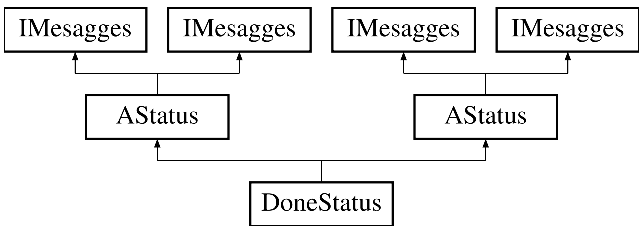
Implements ILoader.

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

- bonus/lib/DLLoader.hpp
- lib/DLLoader.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]
```

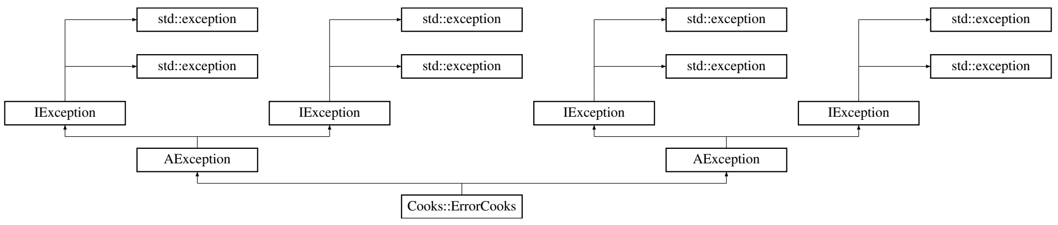
Implements AStatus.

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

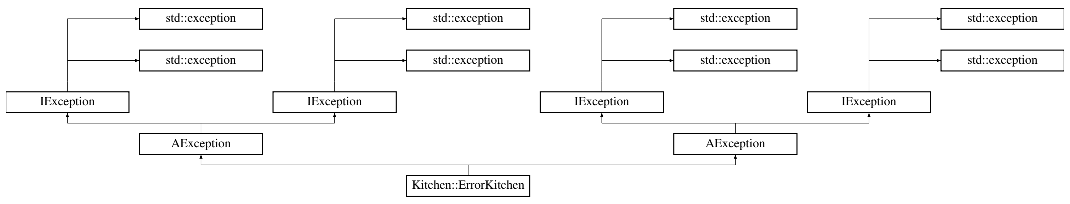
- **AException** (const std::string &type, const std::string &message)
- const char ∗ what () const noexcept override
- std::string getType () const noexcept override
- std::string getMessage () const noexcept override
- std::string getFormattedMessage () const noexcept override
- **AException** (const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

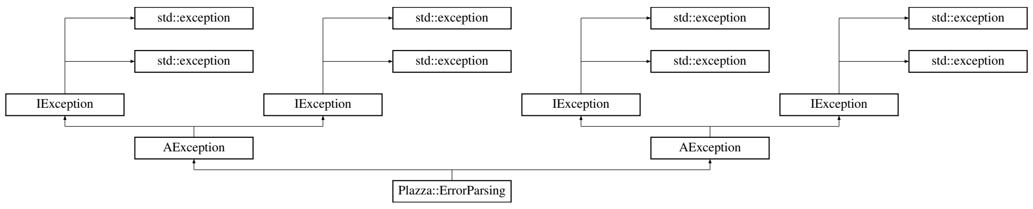
- **AException** (const std::string &type, const std::string &message)
- const char ∗ what () const noexcept override
- std::string getType () const noexcept override
- std::string getMessage () const noexcept override
- std::string getFormattedMessage () const noexcept override
- **AException** (const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

Inheritance diagram for Plazza::ErrorParsing:



Public Member Functions

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

Public Member Functions inherited from AException

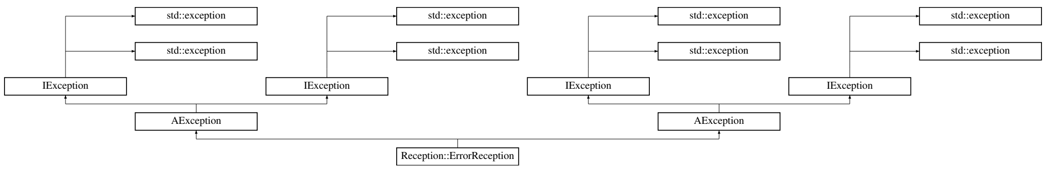
- **AException** (const std::string &type, const std::string &message)
- const char ∗ what () const noexcept override
- std::string getType () const noexcept override
- std::string getMessage () const noexcept override
- std::string getFormattedMessage () const noexcept override
- **AException** (const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

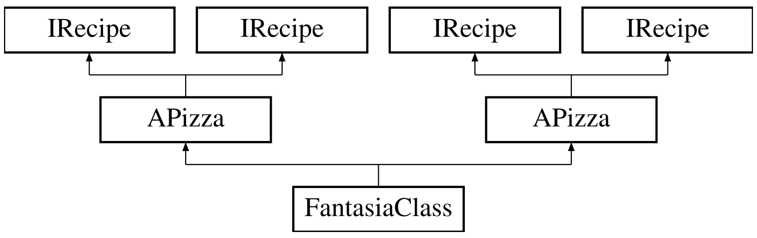
- **AException** (const std::string &type, const std::string &message)
- const char ∗ what () const noexcept override
- std::string getType () const noexcept override
- std::string getMessage () const noexcept override
- std::string getFormattedMessage () const noexcept override
- **AException** (const 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 > ingridient) override
- void serve () override
- **FantasiaClass** (int number)
- void cook (int cookTime) override
- std::shared_ptr< Ingridient > prepare (int number, std::shared_ptr< Ingridient > ingridient) override
- void serve () override

Public Member Functions inherited from APizza

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

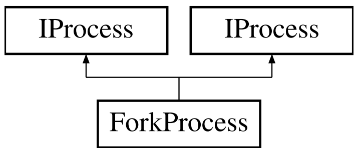
Implements APizza.

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]
```

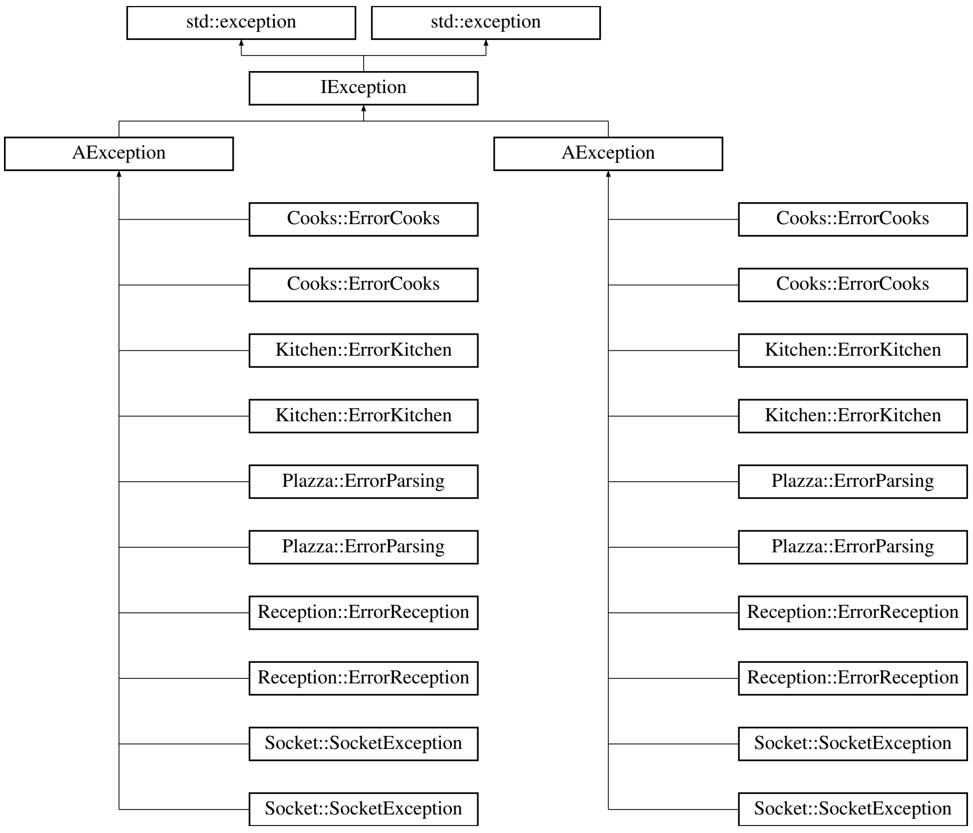
Implements IProcess.

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

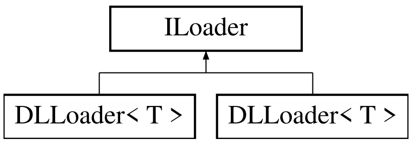
- const char ∗ **what** () const noexcept override=0
- virtual std::string **getType** () const noexcept=0
- virtual std::string **getMessage** () const noexcept=0
- virtual std::string **getFormattedMessage** () const noexcept=0
- const char ∗ **what** () const noexcept override=0
- virtual std::string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

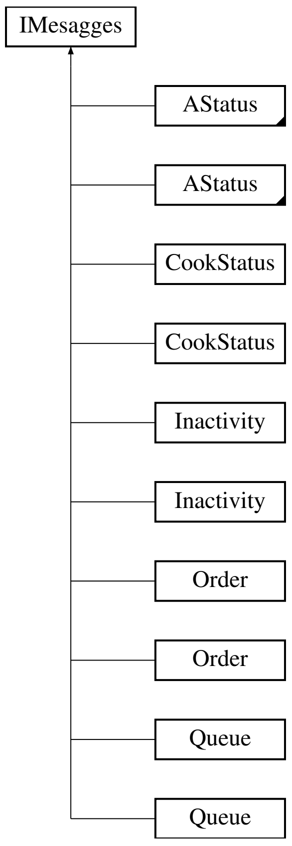
- virtual void ∗ **Open** (const char ∗path, int flag)=0
- virtual void ∗ **Symbol** (const char ∗symbolName)=0
- virtual int **Close** ()=0
- virtual const char ∗ **Error** ()=0
- virtual void ∗ **getHandler** () const =0
- virtual void ∗ **Open** (const char ∗path, int flag)=0
- virtual void ∗ **Symbol**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

Inheritance diagram for IMesagges:



Public Member Functions

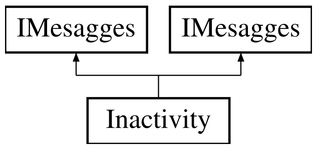
- virtual std::string **pack** (const IMesagges &messages) const =0
- virtual std::shared_ptr< IMesagges > **unpack** (const std::string &data)=0
- virtual MessageType **getType** () const =0
- virtual std::string **typeToString** (MessageType type) const =0
- virtual std::string **pack** (const IMesagges &messages) const =0
- virtual std::shared_ptr< IMesagges > **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/IMesagges.hpp
- common/messages/IMesagges.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 > **_ingridient**

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

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

5.25 ingStat Struct Reference

Public Attributes

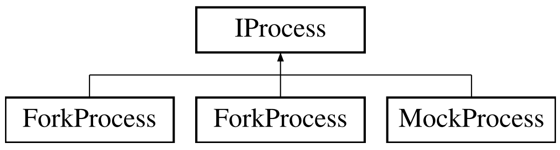
- IngridientType **type**
- int **quantity**

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

- bonus/common/Ingridient.hpp
- common/Ingridient.hpp

5.26 IProcess Class Reference

Inheritance diagram for IProcess:



Public Member Functions

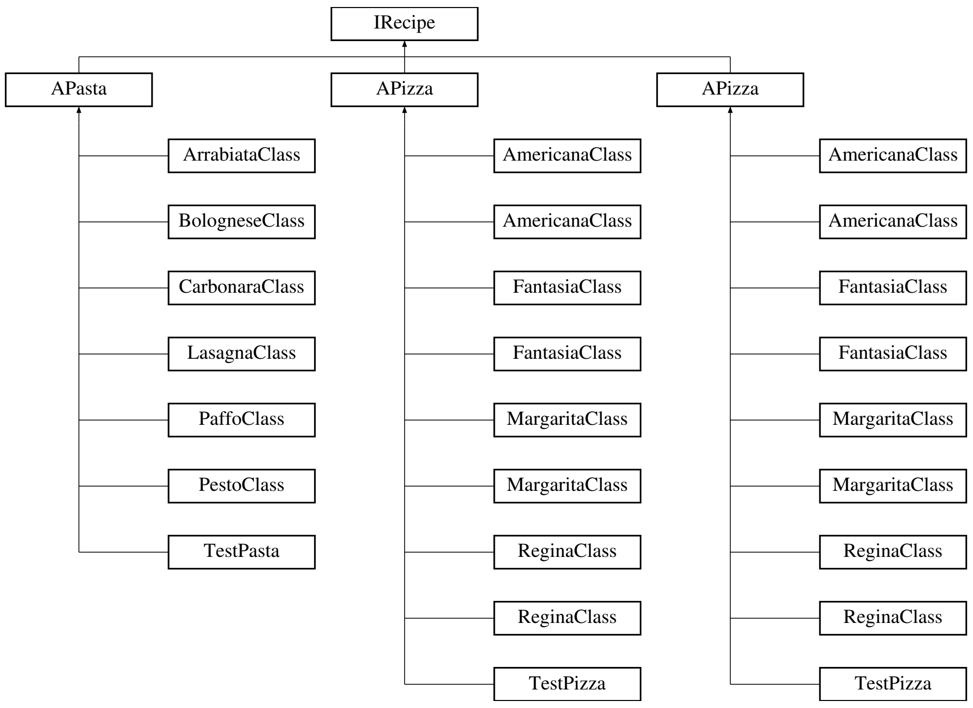
- virtual pid_t **create** (const std::function< void()> &childLogic)=0
- virtual int **wait** ()=0
- virtual void **close** ()=0
- virtual pid_t **getPid** () const =0
- virtual pid_t **create** (const std::function< void()> &childLogic)=0
- virtual int **wait** ()=0
- virtual void **close** ()=0
-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

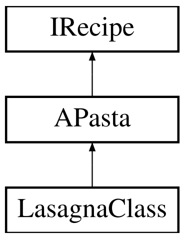
- int **_ID**
- int **_nbCooks**
- int **_cookTime**
- int **_restockTime**
- int **_maxCmd**
- int **_currentOrders**
- pid_t **_pid**
- std::shared_ptr< Ingridient > **_ingridient**
- std::vector< std::shared_ptr< Cooks > > **_cooks**
- std::queue< std::string > **_orderQueue**
- std::mutex **_orderMutex**
- 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]
```

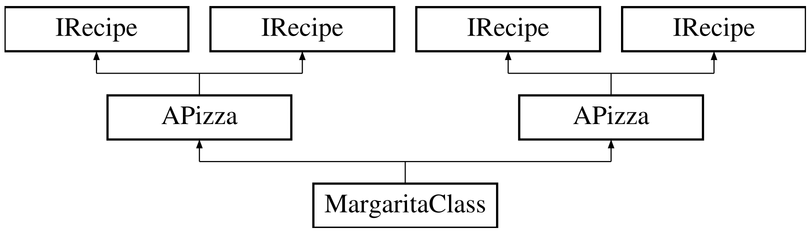
Implements APasta.

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

Public Member Functions inherited from APizza

- **APizza** (int number)
- int getNumber () const override
- void setNumber (int number) override
- **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 > ingridient ) [override], [virtual]
```

Implements APizza.

5.30.1.4 prepare() [2/2]

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

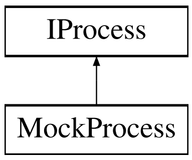
Implements APizza.

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

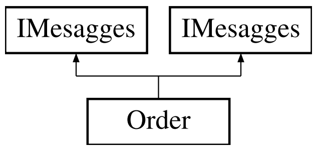
- bool **_shouldFailCreate**
- bool **_shouldFailAccept**
- bool **_shouldFailConnect**
- bool **_isConnected**
- std::string **_lastSentMessage**
- std::queue< std::string > **_receivedMessages**

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 IMesagges.

5.33.1.2 getType() [2/2]

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

Implements IMesagges.

5.33.1.3 pack() [1/2]

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

Implements IMesagges.

5.33.1.4 pack() [2/2]

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

Implements IMesagges.

5.33.1.5 typeToString() [1/2]

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

Implements IMesagges.

5.33.1.6 typeToString() [2/2]

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

Implements IMesagges.

5.33.1.7 unpack() [1/2]

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

Implements IMesagges.

5.33.1.8 unpack() [2/2]

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

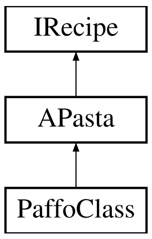
Implements IMesagges.

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]
```

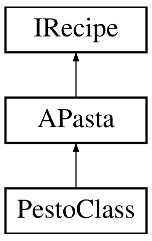
Implements APasta.

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 > ingridient )  [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 Plazza 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

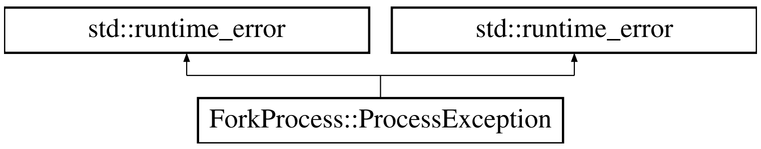
- int **_nbCooks**
- int **_timerCooker**
- int **_timerRestock**
- bool **_debug**
- Reception **_reception**

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

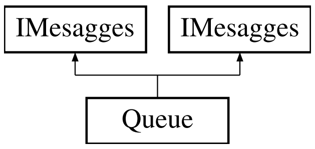
- **ProcessException** (const std::string &msg)
- **ProcessException** (const std::string &msg)

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 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
- **Queue** (int id, int currentOrders)
- 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 **kitchenId**
- int **nbCurrentOrders**

5.38.1 Member Function Documentation

5.38.1.1 getType() [1/2]

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

Implements IMesagges.

5.38.1.2 getType() [2/2]

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

Implements IMesagges.

5.38.1.3 pack() [1/2]

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

Implements IMesagges.

5.38.1.4 pack() [2/2]

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

Implements IMesagges.

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

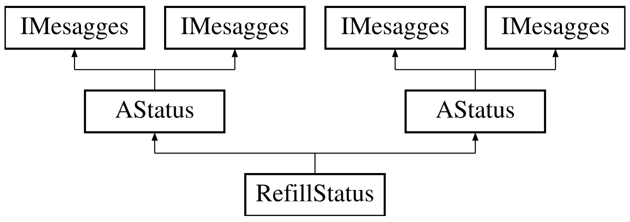
- int **_nbCooks**
- int **_cookTime**
- int **_restockTime**
- int **_nbKitchens**
- bool **_isDebug**
- std::vector< std::shared_ptr< Kitchen > > **_kitchens**
- std::unordered_map< int, std::shared_ptr< Socket > > **_kitchenSockets**
- std::mutex **_kitchensMutex**
- std::atomic< bool > **_isRunning**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

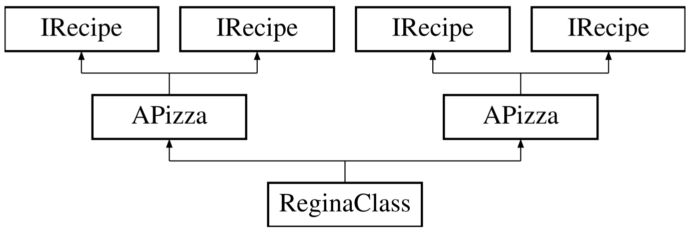
Implements AStatus.

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

Public Member Functions inherited from APizza

- **APizza** (int number)
- int getNumber () const override
- void setNumber (int number) override
- **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 > ingridient )  [override], [virtual]
```

Implements APizza.

5.41.1.4 prepare() [2/2]

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

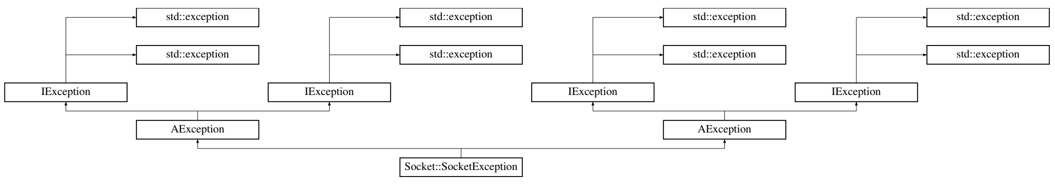
- int **_serverFd**
- int **_clientFd**
- struct sockaddr_un **_addr**
- std::string **_sockPath**
- bool **_isServer**
- bool **_isConnected**

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

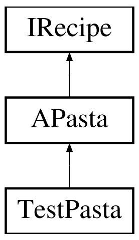
- **AException** (const std::string &type, const std::string &message)
- const char ∗ what () const noexcept override
- std::string getType () const noexcept override
- std::string getMessage () const noexcept override
- std::string getFormattedMessage () const noexcept override
- **AException** (const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

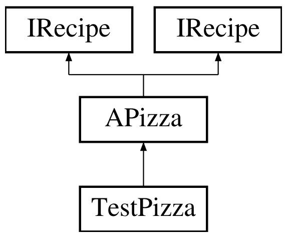
Implements APasta.

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 > ingridient )  [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

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

#ifndef APASTA_HPP_
#define APASTA_HPP_

#include "IRecipe.hpp"


class APasta : public IRecipe {
    public:
        APasta(int number);
        virtual ~APasta() override = default;
        virtual void cook(int cookTime) override = 0;
        virtual std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient>
ingridient) override = 0;
        virtual void serve() override = 0;

        /* Getter */
        int getNumber() const override;

        /* Setter */
        void setNumber(int number) override;

    private:
        int _size;
        int _number;
};

#endif /* !APASTA_HPP_ */
```

6.2 Arrabiata.hpp

```
/*
** EPITECH PROJECT, 2025
** The Plazza
** File description:
** Arrabiata
*/

#include "../../../common/APasta.hpp"

#ifndef ARRABIATA_HPP_
    #define ARRABIATA_HPP_

class ArrabiataClass : public APasta {
    public:
        ArrabiataClass(int number);
        ~ArrabiataClass() override;

        /* Method */
        void cook(int cookTime) override;
        std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient> ingridient) override;
        void serve() override;

    protected:
    private:
};

#endif /* !ARRABIATA_HPP_ */
```

6.3 Bolognese.hpp

```
/*
** EPITECH PROJECT, 2025
** The Plazza
** File description:
** Boloss
*/

#include "../../../common/APasta.hpp"

#ifndef BOLOGNESE_HPP_
    #define BOLOGNESE_HPP_

class BologneseClass : public APasta {
    public:
        BologneseClass(int number);
        ~BologneseClass() override;

        /* Method */
        void cook(int cookTime) override;
        std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient> ingridient) override;
        void serve() override;

    private:
};

#endif /* !BOLOGNESE_HPP_ */
```

6.4 Carbonara.hpp

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

#include "../../../common/APasta.hpp"

#ifndef CARBONARA_HPP_
    #define CARBONARA_HPP_

class CarbonaraClass : public APasta {
    public:
        CarbonaraClass(int number);
        ~CarbonaraClass() override;

        /* Method */
        void cook(int cookTime) override;
        std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient> ingridient) override;
        void serve() override;

    private:
};

#endif /* !CARBONAR_HPP_ */
```

6.5 Lasagna.hpp

```
/*
```

```
** EPITECH PROJECT, 2025
** The Plazza
** File description:
** Lasagna
*/

#include "../../../common/APasta.hpp"

#ifndef LASAGNA_HPP_
    #define LASAGNA_HPP_

class LasagnaClass : public APasta {
    public:
        LasagnaClass(int number);
        ~LasagnaClass() override;

        /* Method */
        void cook(int cookTime) override;
        std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient> ingridient)
override;
        void serve() override;

    private:
};

#endif /* !LASAGNA_HPP_ */
```

6.6 Paffo.hpp

```
/*
** EPITECH PROJECT, 2025
** The Plazza
** File description:
** Paffo
*/

#include "../../../common/APasta.hpp"

#ifndef PAFFO_HPP_
    #define PAFFO_HPP_

class PaffoClass : public APasta {
    public:
        PaffoClass(int number);
        ~PaffoClass() override;

        /* Method */
        void cook(int cookTime) override;
        std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient> ingridient)
override;
        void serve() override;

    private:
};

#endif /* !PAFFO_HPP_ */
```

6.7 Pesto.hpp

```
/*
** EPITECH PROJECT, 2025
** Friteuse
** File description:
** Pesto
*/

#include "../../../common/APasta.hpp"

#ifndef PESTO_HPP_
    #define PESTO_HPP_

class PestoClass : public APasta {
    public:
        PestoClass(int number);
        ~PestoClass() override;

        /* Method */
        void cook(int cookTime) override;
        std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient> ingridient)
override;
        void serve() override;

    private:
};

#endif /* !PESTO_HPP_ */
```

6.8 AException.hpp

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

#ifndef AEXEPTION_HPP_
    #define AEXEPTION_HPP_

#include "IException.hpp"
#include <string>

class AException : public IException {
    public:
        AException(const std::string& type, const std::string& message)
        : _message(message), _type(type) {}
        virtual ~AException() noexcept = default;

        const char* what() const noexcept override {
            return getFormattedMessage().c_str();
        }

        std::string getType() const noexcept override {
            return _type;
        }

        std::string getMessage() const noexcept override {
            return _message;
        }

        std::string getFormattedMessage() const noexcept override {
            return "\033[1;31m[" + _type + "]\033[0m " + _message;
        }

    private:
        std::string _message;
        std::string _type;
};

#endif /* !AEXEPTION_HPP_ */
```

6.9 AException.hpp

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

#ifndef AEXEPTION_HPP_
    #define AEXEPTION_HPP_

#include "IException.hpp"
#include <string>

class AException : public IException {
    public:
        AException(const std::string& type, const std::string& message)
        : _message(message), _type(type) {}
        virtual ~AException() noexcept = default;

        const char* what() const noexcept override {
            return getFormattedMessage().c_str();
        }

        std::string getType() const noexcept override {
            return _type;
        }

        std::string getMessage() const noexcept override {
            return _message;
        }

        std::string getFormattedMessage() const noexcept override {
            return "\033[1;31m[" + _type + "]\033[0m " + _message;
        }

    private:
        std::string _message;
        std::string _type;
};

#endif /* !AEXEPTION_HPP_ */
```

6.10 APizza.hpp

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

#ifndef APIZZA_HPP_
#define APIZZA_HPP_

#include "IRecipe.hpp"


class APizza : public IRecipe {
    public:
        APizza(int number);
        virtual ~APizza() override = default;
        virtual void cook(int cookTime) override = 0;
        virtual std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient>
ingridient) override = 0;
        virtual void serve() override = 0;

        /* Getter */
        int getNumber() const override;

        /* Setter */
        void setNumber(int number) override;
    private:
        int _number;

};

#endif /* !APIZZA_HPP_ */
```

6.11 APizza.hpp

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

#ifndef APIZZA_HPP_
#define APIZZA_HPP_

#include "IRecipe.hpp"

enum PizzaType
{
    Nothing = 0,
    Regina = 1,
    Margarita = 2,
    Americana = 4,
    Fantasia = 8
};


class APizza : public IRecipe {
    public:
        APizza(int number);
        virtual ~APizza() override = default;
        virtual void cook(int cookTime) override = 0;
        virtual std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient>
ingridient) override = 0;
        virtual void serve() override = 0;

        /* Getter */
        int getNumber() const override;

        /* Setter */
        void setNumber(int number) override;
    private:
        int _number;

};

#endif /* !APIZZA_HPP_ */
```

6.12 IException.hpp

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

#include <exception>
#include <string>

#ifndef IEXEPTION_HPP_
    #define IEXEPTION_HPP_

class IException : public std::exception {
    public:
        virtual ~IException() noexcept = default;
        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;
};

#endif /* !IEXEPTION_HPP_ */
```

6.13 IException.hpp

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

#include <exception>
#include <string>

#ifndef IEXEPTION_HPP_
    #define IEXEPTION_HPP_

class IException : public std::exception {
    public:
        virtual ~IException() noexcept = default;
        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;
};

#endif /* !IEXEPTION_HPP_ */
```

6.14 Ingridient.hpp

```
/*
```

```cpp
** EPITECH PROJECT, 2025
** B-CCP-400-NAN-4-1-theplazza-albane.merian
** File description:
** Ingridient
*/

#include <vector>
#include <string>
#include <sstream>
#include <map>

#ifndef INGRIDIENT_HPP_
#define INGRIDIENT_HPP_

enum IngridientType
{
    DOUGH = 0,
    TOMATO = 1,
    CHEESE = 2,
    HAM = 3,
    MUSHROOM = 4,
    STEAK = 5,
    EGGPLANT = 6,
    GOAT_CHEESE = 7,
    CHEF_LOVE = 8,
    EGG = 9,
    BACON = 10,
    BASIL = 11,
    PEPPER = 12
};

struct ingStat {
    IngridientType type;
    int quantity;
};

class Ingridient {
    public:
        Ingridient();
        ~Ingridient() = default;
        std::vector<ingStat> fridgeStatus();

        /* Getter */
        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;

        /* Setter */
        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);

        /* Packing/Unpacking methods */
        std::string packIngredients() const;
        static std::map<IngridientType, int> unpackIngredients(const std::string& packedData);
        std::shared_ptr<Ingridient> operator=(const std::vector<ingStat> &ingStat);
    private:
        int _dough;
        int _tomato;
        int _cheese;
        int _ham;
        int _mushroom;
        int _steak;
        int _eggplant;
        int _goatCheese;
        int _chefLove;
        int _egg;
        int _bacon;
        int _basil;
        int _pepper;
        std::vector<ingStat> _ingridient;
};

#endif /* !INGRIDIENT_HPP_ */
```

6.15 Ingridient.hpp

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

#include <vector>
#include <string>
#include <memory>
#include <sstream>
#include <map>

#ifndef INGRIDIENT_HPP_
#define INGRIDIENT_HPP_

enum IngridientType
{
    DOUGH = 0,
    TOMATO = 1,
    CHEESE = 2,
    HAM = 3,
    MUSHROOM = 4,
    STEAK = 5,
    EGGPLANT = 6,
    GOAT_CHEESE = 7,
    CHEF_LOVE = 8,
    EGG = 9,
    BACON = 10,
    BASIL = 11,
    PEPPER = 12
};

struct ingStat {
    IngridientType type;
    int quantity;
};

class Ingridient {
    public:
        Ingridient();
        ~Ingridient() = default;
        std::vector<ingStat> fridgeStatus();

        /* Getter */
        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;

        /* Setter */
        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);

        /* Packing/Unpacking methods */
        std::string packIngredients() const;
        static std::map<IngridientType, int> unpackIngredients(const std::string& packedData);
        std::shared_ptr<Ingridient> operator=(const std::vector<ingStat> &ingStat);
    private:
        int _dough;
        int _tomato;
        int _cheese;
        int _ham;
        int _mushroom;
        int _steak;
        int _eggplant;
        int _goatCheese;
        int _chefLove;
        int _egg;
        int _bacon;
        int _basil;
        int _pepper;
        std::vector<ingStat> _ingridient;
};

#endif /* !INGRIDIENT_HPP_ */
```

6.16 IRecipe.hpp

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


#include <memory>
#include "Ingridient.hpp"

#ifndef IRECIPE_HPP_
#define IRECIPE_HPP_

enum Size
{
    Zero = 0,
    S = 1,
    M = 2,
    L = 4,
    XL = 8,
    XXL = 16
};

enum RecipyType
{
    /* Pizza types*/
    Nothing = 0,
    Regina = 1,
    Margarita = 2,
    Americana = 4,
    Fantasia = 8,

    /* Pasta Types */
    Carbonara = 10,
    Pesto = 12,
    Bolognese = 14,
    Arrabiata = 16,
    Paffo = 18,
    Lasagna = 20
};


class IRecipe {
    public:

        virtual ~IRecipe() = default;
        virtual void cook(int cookTime) = 0;
        virtual std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient>
ingridient) = 0;
        virtual void serve() = 0;

        /* Getter */
        virtual int getNumber() const = 0;
        /* Setter */
        virtual void setNumber(int number) = 0;

};

#endif /* !IRECEIPY_HPP_ */
```

6.17 IRecipe.hpp

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


#include <memory>
#include "Ingridient.hpp"

#ifndef IRECIPE_HPP_
#define IRECIPE_HPP_

enum Size
{
    Zero = 0,
    S = 1,
    M = 2,
    L = 4,
    XL = 8,
    XXL = 16
};


class IRecipe {
    public:

        virtual ~IRecipe() = default;
        virtual void cook(int cookTime) = 0;
        virtual std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient>
ingridient) = 0;
        virtual void serve() = 0;

        /* Getter */
        virtual int getNumber() const = 0;
        /* Setter */
        virtual void setNumber(int number) = 0;

};

#endif /* !IRECEIPY_HPP_ */
```

6.18 AStatus.hpp

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


#include <memory>

#include "../Ingridient.hpp"
#include "IMesagges.hpp"

#ifndef ASTATUS_HPP_
#define ASTATUS_HPP_

class AStatus : public IMesagges {
    public:
        int kitchenId;
        std::vector<ingStat> status;

        /* Constrcutor */
        AStatus(int id, std::vector<ingStat> status);
        ~AStatus() override = default;

        /* Pack */
        std::string pack(const IMesagges &messages) const override;

        /* Unpack */
        virtual std::shared_ptr<IMesagges> unpack(const std::string &data) override = 0;

        /* Ox to str */
        std::string typeToString(MessageType type) const override;

        /* Virtual function that needs to be override */
        virtual MessageType getType() const override = 0;
    protected:
    private:
};

#endif /* !ASTATUS_HPP_ */
```

6.19 AStatus.hpp

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


#include <memory>

#include "../Ingridient.hpp"
#include "IMesagges.hpp"

#ifndef ASTATUS_HPP_
#define ASTATUS_HPP_

class AStatus : public IMesagges {
    public:
        int kitchenId;
        std::vector<ingStat> status;

        /* Constrcutor */
        AStatus(int id, std::vector<ingStat> status);
        ~AStatus() override = default;

        /* Pack */
        std::string pack(const IMesagges &messages) const override;

        /* Unpack */
        virtual std::shared_ptr<IMesagges> unpack(const std::string &data) override = 0;

        /* Ox to str */
        std::string typeToString(MessageType type) const override;

        /* Virtual function that needs to be override */
        virtual MessageType getType() const override = 0;
    protected:
    private:
};

#endif /* !ASTATUS_HPP_ */
```

6.20 CookStatus.hpp

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

#include <vector>
#include <memory>

#include "IMesagges.hpp"

#ifndef COOKSTATUS_HPP_
#define COOKSTATUS_HPP_

struct CookStatusData {
    int cookId;
    bool isBusy;
    bool isRestocking;
};

class CookStatus : public IMesagges {
    public:
        int _kitchenId;
        std::vector<CookStatusData> _cooksStatus;

        CookStatus(int kitchenID, const std::vector<CookStatusData> &cooksStatus);
        ~CookStatus() override = default;

        /* Pack */
        std::string pack(const IMesagges &messages) const override;

        /* Unpack */
        std::shared_ptr<IMesagges> unpack(const std::string &data) override;

        MessageType getType() const override;
        std::string typeToString(MessageType type) const override;
    protected:
    private:
};

#endif /* !COOKSTATUS_HPP_ */
```

6.21 CookStatus.hpp

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

#include <vector>
#include <memory>

#include "IMesagges.hpp"

#ifndef COOKSTATUS_HPP_
#define COOKSTATUS_HPP_

struct CookStatusData {
    int cookId;
    bool isBusy;
    bool isRestocking;
};

class CookStatus : public IMesagges {
    public:
        int _kitchenId;
        std::vector<CookStatusData> _cooksStatus;

        CookStatus(int kitchenID, const std::vector<CookStatusData> &cooksStatus);
        ~CookStatus() override = default;

        /* Pack */
        std::string pack(const IMesagges &messages) const override;

        /* Unpack */
        std::shared_ptr<IMesagges> unpack(const std::string &data) override;

        MessageType getType() const override;
        std::string typeToString(MessageType type) const override;
    protected:
    private:
};

#endif /* !COOKSTATUS_HPP_ */
```

6.22 DoneStatus.hpp

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

#include "AStatus.hpp"
```

```

#ifndef DONESTATUS_HPP_
#define DONESTATUS_HPP_

class DoneStatus : public AStatus {
    public:
        DoneStatus(int id, std::vector<ingStat> status);
        MessageType getType() const override;
        std::shared_ptr<IMesagges> unpack(const std::string &data) override;

    protected:
    private:
};

#endif /* !DONESTATUS_HPP_ */
```

6.23 DoneStatus.hpp

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

#include "AStatus.hpp"

#ifndef DONESTATUS_HPP_
#define DONESTATUS_HPP_

class DoneStatus : public AStatus {
    public:
        DoneStatus(int id, std::vector<ingStat> status);
        MessageType getType() const override;
        std::shared_ptr<IMesagges> unpack(const std::string &data) override;

    protected:
    private:
};

#endif /* !DONESTATUS_HPP_ */
```

6.24 IMesagges.hpp

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

#include <string>
#include <memory>


#ifndef IMESAGGES_HPP_
#define IMESAGGES_HPP_


enum class MessageType : uint8_t {
    Order = 0x01,
    Status = 0x02,
    Inactivity = 0x03,
    Refill = 0x04,
    Queue = 0x05,
    CookStatus = 0x06,
};


class IMesagges {
    public:
        virtual ~IMesagges() = default;
        virtual std::string pack(const IMesagges &messages) const = 0;
        virtual std::shared_ptr<IMesagges> unpack(const std::string &data) = 0;
        virtual MessageType getType() const = 0;
        virtual std::string typeToString(MessageType type) const = 0;

    protected:
    private:
};

#endif /* !IMESAGGES_HPP_ */
```

6.25 IMesagges.hpp

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

#include <string>
#include <memory>


#ifndef IMESAGGES_HPP_
#define IMESAGGES_HPP_


enum class MessageType : uint8_t {
    Order = 0x01,
    Status = 0x02,
    Inactivity = 0x03,
    Refill = 0x04,
    Queue = 0x05,
    CookStatus = 0x06,
};


class IMesagges {
    public:
        virtual ~IMesagges() = default;
        virtual std::string pack(const IMesagges &messages) const = 0;
        virtual std::shared_ptr<IMesagges> unpack(const std::string &data) = 0;
        virtual MessageType getType() const = 0;
        virtual std::string typeToString(MessageType type) const = 0;

    protected:
    private:
};

#endif /* !IMESAGGES_HPP_ */
```

6.26 Inactivity.hpp

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


#include <memory>

#include "IMesagges.hpp"

#ifndef INACTIVITY_HPP_
#define INACTIVITY_HPP_

class Inactivity : public IMesagges {
    public:
        int id;

        /* Constructor */
        Inactivity(int kitchenId);
        ~Inactivity() override = default;

        /* Pack */
        std::string pack(const IMesagges &messages) const override;

        /* Unpack */
        std::shared_ptr<IMesagges> unpack(const std::string &data) override;

        MessageType getType() const override;
        std::string typeToString(MessageType type) const override;
    protected:
```

```
    private:
};

#endif /* !INACTIVITY_HPP_ */
```

6.27 Inactivity.hpp

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


#include <memory>

#include "IMesagges.hpp"

#ifndef INACTIVITY_HPP_
#define INACTIVITY_HPP_

class Inactivity : public IMesagges {
    public:
        int id;

        /* Constructor */
        Inactivity(int kitchenId);
        ~Inactivity() override = default;

        /* Pack */
        std::string pack(const IMesagges &messages) const override;

        /* Unpack */
        std::shared_ptr<IMesagges> unpack(const std::string &data) override;

        MessageType getType() const override;
        std::string typeToString(MessageType type) const override;
    protected:
    private:
};

#endif /* !INACTIVITY_HPP_ */
```

6.28 Order.hpp

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

#include <iostream>
#include <sstream>
#include <regex>
#include <string>

#include "../APizza.hpp"
#include "IMesagges.hpp"


class Order : public IMesagges {
    public:
        RecipyType type;
        Size size;
        int number;

    /* Constructor */
    Order(RecipyType t, Size s, int n);
    ~Order() override = default;

    /* Pack order Message */
    std::string pack(const IMesagges &order) const override;

    /* Unpack Order */
    std::shared_ptr<IMesagges> unpack(const std::string &data) override;

    MessageType getType() const override;
    std::string typeToString(MessageType type) const override;
};
```

6.29 Order.hpp

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

#include <iostream>
#include <sstream>
#include <regex>
#include <string>

#include "../APizza.hpp"
#include "IMesagges.hpp"


class Order : public IMesagges {
    public:
        PizzaType type;
        Size size;
        int number;

    /* Constructor */
    Order(PizzaType t, Size s, int n);
    ~Order() override = default;

    /* Pack order Message */
    std::string pack(const IMesagges &order) const override;

    /* Unpack Order */
    std::shared_ptr<IMesagges> unpack(const std::string &data) override;

    MessageType getType() const override;
    std::string typeToString(MessageType type) const override;
};
```

6.30 Queue.hpp

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

#include <memory>

#include "IMesagges.hpp"

#ifndef QUEUE_HPP_
#define QUEUE_HPP_

class Queue : public IMesagges {
    public:
        int kitchenId;
        int nbCurrentOrders;

        /* Constructor */
        Queue(int id, int currentOrders);
        ~Queue() override = default;

        /* Pack */
        std::string pack(const IMesagges &messages) const override;

        /* Unpack */
        std::shared_ptr<IMesagges> unpack(const std::string &data) override;

        MessageType getType() const override;
        std::string typeToString(MessageType type) const override;
    protected:
    private:
};

#endif /* !QUEUE_HPP_ */
```

6.31 Queue.hpp

```
/*
```

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

#include <memory>

#include "IMesagges.hpp"

#ifndef QUEUE_HPP_
#define QUEUE_HPP_

class Queue : public IMesagges {
    public:
        int kitchenId;
        int nbCurrentOrders;

        /* Constructor */
        Queue(int id, int currentOrders);
        ~Queue() override = default;

        /* Pack */
        std::string pack(const IMesagges &messages) const override;

        /* Unpack */
        std::shared_ptr<IMesagges> unpack(const std::string &data) override;

        MessageType getType() const override;
        std::string typeToString(MessageType type) const override;
    protected:
    private:
};

#endif /* !QUEUE_HPP_ */
```

6.32 RefillStatus.hpp

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

#include "AStatus.hpp"

#ifndef REFILLSTATUS_HPP_
#define REFILLSTATUS_HPP_

class RefillStatus : public AStatus {
    public:
        RefillStatus(int id, std::vector<ingStat> status);
        MessageType getType() const override;
        std::shared_ptr<IMesagges> unpack(const std::string &data) override;

    protected:
    private:
};

#endif /* !REFILLSTATUS_HPP_ */
```

6.33 RefillStatus.hpp

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

#include "AStatus.hpp"

#ifndef REFILLSTATUS_HPP_
#define REFILLSTATUS_HPP_

class RefillStatus : public AStatus {
    public:
        RefillStatus(int id, std::vector<ingStat> status);
        MessageType getType() const override;
        std::shared_ptr<IMesagges> unpack(const std::string &data) override;

    protected:
    private:
};

#endif /* !REFILLSTATUS_HPP_ */
```

6.34 ForkProcess.hpp

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

#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdexcept>
#include <functional>
#include <iostream>

#include "IProcess.hpp"

#ifndef FORKPROCESS_HPP_
#define FORKPROCESS_HPP_

class ForkProcess : public IProcess {
    class ProcessException : public std::runtime_error {
        public:
            explicit ProcessException(const std::string& msg) : std::runtime_error(msg) {}
    };
    public:
        ForkProcess();
        ~ForkProcess() override;

        /* Override Methods */
        pid_t create(const std::function<void()> &childLogic) override;
        int wait() override;
        void close() override;
        pid_t getPid() const override;

    protected:
    private:
        pid_t _pid;
};

#endif /* !FORKPROCESS_HPP_ */
```

6.35 ForkProcess.hpp

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

#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdexcept>
#include <functional>
#include <iostream>

#include "IProcess.hpp"

#ifndef FORKPROCESS_HPP_
#define FORKPROCESS_HPP_

class ForkProcess : public IProcess {
    class ProcessException : public std::runtime_error {
        public:
            explicit ProcessException(const std::string& msg) : std::runtime_error(msg) {}
    };
    public:
        ForkProcess();
        ~ForkProcess() override;

        /* Override Methods */
        pid_t create(const std::function<void()> &childLogic) override;
        int wait() override;
        void close() override;
        pid_t getPid() const override;

    protected:
    private:
        pid_t _pid;
};

#endif /* !FORKPROCESS_HPP_ */
```

6.36 IProcess.hpp

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

#include <sys/types.h>
#include <sys/wait.h>

#include <iostream>
#include <functional>

#ifndef IPROCESS_HPP_
#define IPROCESS_HPP_

class IProcess {
    public:
        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 ~IProcess() = default;
    protected:
    private:
};

#endif /* !IPROCESS_HPP_ */
```

6.37 IProcess.hpp

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

#include <sys/types.h>
#include <sys/wait.h>

#include <iostream>
#include <functional>

#ifndef IPROCESS_HPP_
#define IPROCESS_HPP_

class IProcess {
    public:
        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 ~IProcess() = default;
    protected:
    private:
};

#endif /* !IPROCESS_HPP_ */
```

6.38 Socket.hpp

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

#ifndef SOCKET_HPP_
#define SOCKET_HPP_

#include <sys/socket.h>
#include <sys/un.h>
#include <string>
#include <unistd.h>
#include <memory>
#include "AException.hpp"

class Socket {

    public:
        class SocketException : public AException {
            public:
                SocketException(const std::string &message);
        };

        Socket();
        ~Socket();

        // Server operations
        void createServer(const std::string &sockPath);
        void acceptClient();
        void closeServer();

        // Client operations
        void connectToServer(const std::string &sockPath);
        void closeClient();

        // Common operations
        ssize_t send(const std::string &message);
        std::string receive(size_t size = 1024);
        bool isConnected() const;

        // Operators
        Socket& operator«(const std::string &message);
        Socket& operator»(std::string &message);

    private:
        int _serverFd;
        int _clientFd;
        struct sockaddr_un _addr;
        std::string _sockPath;
        bool _isServer;
        bool _isConnected;
};

#endif /* !SOCKET_HPP_ */
```

6.39 Socket.hpp

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

#ifndef SOCKET_HPP_
#define SOCKET_HPP_

#include <sys/socket.h>
#include <sys/un.h>
#include <string>
#include <unistd.h>
#include <memory>
#include "AException.hpp"

class Socket {

    public:
        class SocketException : public AException {
            public:
                SocketException(const std::string &message);
        };

        Socket();
        ~Socket();

        // Server operations
        void createServer(const std::string &sockPath);
        void acceptClient();
        void closeServer();

        // Client operations
        void connectToServer(const std::string &sockPath);
        void closeClient();

        // Common operations
        ssize_t send(const std::string &message);
        std::string receive(size_t size = 1024);
        bool isConnected() const;

        // Operators
        Socket& operator«(const std::string &message);
        Socket& operator»(std::string &message);

    private:
        int _serverFd;
        int _clientFd;
        struct sockaddr_un _addr;
        std::string _sockPath;
        bool _isServer;
        bool _isConnected;
};

#endif /* !SOCKET_HPP_ */
```

6.40 DLLoader.hpp

```
/*
** EPITECH PROJECT, 2025
** arcade
** File description:
** DLLoader
*/

#ifndef DLLOADER_HPP_
#define DLLOADER_HPP_

#include <dlfcn.h>
#include <iostream>
#include <ostream>
#include "ILoader.hpp"

template <typename T>

class DLLoader  : public ILoader {
    private:
        void *_handler = nullptr;

    public:
        ~DLLoader() = default;

        void *getHandler() const override {
            return _handler;
        };
        void *Open(const char *path, int flag) override {
            _handler = dlopen(path, flag);
            return _handler;
        };
        void *Symbol(const char *symbolName) override {
            void *symbol = dlsym(_handler, symbolName);
            const char *error = dlerror();
            if (error) {
                std::cerr « "dlerror: " « error « std::endl;
                return nullptr;
            }
            return symbol;
        };
        T getSymbol(const char *symbolName) {
            return reinterpret_cast<T>(dlsym(_handler, symbolName));
        };
        int Close() override{
            if (_handler == nullptr)
                return -1;
            return dlclose(_handler);
        };
        const char *Error() override {
            return dlerror();
        };
};

#endif /* !DLLOADER_HPP_ */
```

6.41 DLLoader.hpp

```
/*
** EPITECH PROJECT, 2025
** arcade
** File description:
** DLLoader
*/

#ifndef DLLOADER_HPP_
#define DLLOADER_HPP_

#include <dlfcn.h>
#include <iostream>
#include <ostream>
#include "ILoader.hpp"

template <typename T>

class DLLoader  : public ILoader {
    private:
        void *_handler = nullptr;

    public:
        ~DLLoader() = default;

        void *getHandler() const override {
            return _handler;
        };
        void *Open(const char *path, int flag) override {
            _handler = dlopen(path, flag);
            return _handler;
        };
        void *Symbol(const char *symbolName) override {
            void *symbol = dlsym(_handler, symbolName);
            const char *error = dlerror();
            if (error) {
                std::cerr « "dlerror: " « error « std::endl;
                return nullptr;
            }
            return symbol;
        };
        T getSymbol(const char *symbolName) {
            return reinterpret_cast<T>(dlsym(_handler, symbolName));
        };
        int Close() override{
            if (_handler == nullptr)
                return -1;
            return dlclose(_handler);
        };
        const char *Error() override {
            return dlerror();
        };
};

#endif /* !DLLOADER_HPP_ */
```

6.42 ILoader.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** ILoader
*/

#ifndef ILoader_HPP_
#define ILoader_HPP_


```

```
class ILoader {
    public:
        ~ILoader() = default;

        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;

    protected:
    private:
};

#endif /* !ILoader_HPP_ */
```

6.43 ILoader.hpp

```
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-raytracer-albane.merian
** File description:
** ILoader
*/

#ifndef ILoader_HPP_
#define ILoader_HPP_


class ILoader {
    public:
        ~ILoader() = default;

        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;

    protected:
    private:
};

#endif /* !ILoader_HPP_ */
```

6.44 Cooks.hpp

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

#include <iostream>
#include <memory>
#include <mutex>
#include <vector>

#include "../../common/AException.hpp"
#include "../../common/Ingridient.hpp"
#include "../../common/IRecipe.hpp"
#include "../../lib/DLLoader.hpp"

#ifndef COOKS_HPP_
    #define COOKS_HPP_

class Cooks {

    class ErrorCooks : public AException {
        public:
            ErrorCooks(const std::string &message);
    };

    public:
        Cooks(std::shared_ptr<Ingridient> ingridient, int id,
            int cookTime, int restockTime);
        ~Cooks() = default;

        /* Method */
        std::shared_ptr<Ingridient> startOrder(std::shared_ptr<Ingridient> ingridient,
std::vector<std::string> order);
        bool hasEnoughIngredients(const std::string &orderData, std::shared_ptr<Ingridient>
ingridient);
        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);
        /* Getter */
        int getID() const;
        bool isBusy() const;
        bool isRestocking() const;

        /* Setter */
        void setIsBusy(bool isBusy);
        /* Loader Pluggins */
        std::string toString(RecipyType type);
        std::shared_ptr<IRecipe> findAndLoadPlugin(const std::string &pizzaType, int number);
    private:
        int _ID;
        int _cookTime;
        int _restockTime;
        std::shared_ptr<Ingridient> _ingridient;
        std::mutex _statusMutex;
        bool _isBusy;
        bool _isRestocking;
};

#endif /* !COOKS_HPP_ */
```

6.45 Cooks.hpp

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

#include <iostream>
#include <memory>
#include <mutex>
#include <vector>

#include "../../common/AException.hpp"
#include "../../common/Ingridient.hpp"
#include "../../common/IRecipe.hpp"
#include "../../lib/DLLoader.hpp"

#include "../../common/APizza.hpp"

#ifndef COOKS_HPP_
    #define COOKS_HPP_

class Cooks {
    public:
    class ErrorCooks : public AException {
        public:
            ErrorCooks(const std::string &message);
    };


        Cooks(std::shared_ptr<Ingridient> ingridient, int id,
            int cookTime, int restockTime);
        ~Cooks() = default;

        /* Method */
        std::shared_ptr<Ingridient> startOrder(std::shared_ptr<Ingridient> ingridient,
std::vector<std::string> order);
        bool hasEnoughIngredients(const std::string &orderData, std::shared_ptr<Ingridient>
ingridient);

        std::shared_ptr<IRecipe> loadPlugin(const std::string &path, int number);
        std::string getType(const std::string& path, DLLoader<IRecipe> loader);
        /* Getter */
        int getID() const;
        bool isBusy() const;
        bool isRestocking() const;

        /* Setter */
        void setIsBusy(bool isBusy);
        /* Loader Pluggins */
        std::string toString(PizzaType type);
        std::shared_ptr<IRecipe> findAndLoadPlugin(const std::string &pizzaType, int number);
    private:
        int _ID;
        int _cookTime;
        int _restockTime;
        std::shared_ptr<Ingridient> _ingridient;
        std::mutex _statusMutex;
        bool _isBusy;
        bool _isRestocking;
};

#endif /* !COOKS_HPP_ */
```

6.46 Kitchen.hpp

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

#ifndef KITCHEN_HPP_
#define KITCHEN_HPP_

#include <queue>
#include <mutex>
#include <vector>
#include <thread>
#include <condition_variable>
#include <chrono>

#include "../cooks/Cooks.hpp"
#include "../../common/processes/IProcess.hpp"
#include "../../common/Socket.hpp"

#include "../../common/AException.hpp"

class Kitchen {

    class ErrorKitchen : public AException {
        public:
            ErrorKitchen(const std::string &message);
    };

    public:
        Kitchen(int id, int nbCooks, int cookTime, int restockTime, bool debug);
        ~Kitchen();
        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);

        // Process management
        void setProcess(std::shared_ptr<IProcess> process) {
            _process = process;
        }

        /* Setter */
        void setCurrentOrders(int currentOrders);
        void incrementCurrentOrders(int amount);
        /* Getter */
        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;

        /* Send Messages */
        void sendQueueStatMessage();
        void sendDoneMessage();
        void sendRefillMessage();
        void sendInactive();
        void sendCookStatus();
    protected:
    private:
        int _ID;
        int _nbCooks;
        int _cookTime;
        int _restockTime;
        int _maxCmd;
        int _currentOrders;
        pid_t _pid;  // Add this line to track the process ID
        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;
};

std::ostream& operator<<(std::ostream& os, const Kitchen& kitchen);

#endif /* !KITCHEN_HPP_ */
```

6.47 Kitchen.hpp

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

#ifndef KITCHEN_HPP_
#define KITCHEN_HPP_

#include <queue>
#include <mutex>
#include <vector>
#include <thread>
#include <condition_variable>
#include <chrono>

#include "../cooks/Cooks.hpp"
#include "../../common/processes/IProcess.hpp"
#include "../../common/Socket.hpp"

#include "../../common/AException.hpp"

class Kitchen {

    public:
    class ErrorKitchen : public AException {
        public:
            ErrorKitchen(const std::string &message);
    };

        Kitchen(int id, int nbCooks, int cookTime, int restockTime, bool debug);
        ~Kitchen();
        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);

        // Process management
        void setProcess(std::shared_ptr<IProcess> process) {
            _process = process;
        }

        /* Setter */
        void setCurrentOrders(int currentOrders);
        void incrementCurrentOrders(int amount);
        /* Getter */
        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;

        /* Send Messages */
        void sendQueueStatMessage();
        void sendDoneMessage();
        void sendRefillMessage();
        void sendInactive();
        void sendCookStatus();
    protected:
    private:
        int _ID;
        int _nbCooks;
        int _cookTime;
        int _restockTime;
        int _maxCmd;
        int _currentOrders;
        pid_t _pid;  // Add this line to track the process ID
        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;
};

std::ostream& operator<<(std::ostream& os, const Kitchen& kitchen);

#endif /* !KITCHEN_HPP_ */
```

6.48 Plazza.hpp

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

#ifndef PLAZZA_HPP_
#define PLAZZA_HPP_

#include "reception/Reception.hpp"
#include "../common/AException.hpp"

class Plazza {

    class ErrorParsing : public AException {
        public:
            ErrorParsing(const std::string &message);
    };

    public:
        Plazza();
        ~Plazza();

        void parseCmd(char **av, int ac);
        void orderingLoop();

    private:
        int _nbCooks;
        int _timerCooker;
        int _timerRestock;
        bool _debug;
        Reception _reception;
};
```

```

#endif /* !PLAZZA_HPP_ */
```

6.49 Plazza.hpp

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

#ifndef PLAZZA_HPP_
#define PLAZZA_HPP_

#include "reception/Reception.hpp"
#include "../common/AException.hpp"

class Plazza {

    class ErrorParsing : public AException {
        public:
            ErrorParsing(const std::string &message);
    };

    public:
        Plazza();
        ~Plazza();

        void parseCmd(char **av, int ac);
        void orderingLoop();

    private:
        int _nbCooks;
        int _timerCooker;
        int _timerRestock;
        bool _debug;
        Reception _reception;
};

#endif /* !PLAZZA_HPP_ */
```

6.50 Reception.hpp

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

#ifndef RECEPTION_HPP_
#define RECEPTION_HPP_

#include <thread>
#include <atomic>
#include <mutex>
#include <unordered_map>


#include "../../common/AException.hpp"
#include "../../common/APizza.hpp"
#include "../../common/Socket.hpp"
#include "../kitchen/Kitchen.hpp"


class Reception {

    class ErrorReception : public AException {
        public:
            ErrorReception(const std::string &message);
    };

    public:
        Reception();
        ~Reception();

        /* Getter */
        int getNbKitchens() const;
        std::vector<std::shared_ptr<Kitchen» getKitchens() const;
        std::shared_ptr<Kitchen> getKitchen(int id) const;

        /* Setter */
        void setValues(int nbCooks, int cookTime, int restockTime,
            bool debug = false);

        /* Methods */
        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);

        /* Parser Methods */
        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);

        /* Messefe func handler */
        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);
    protected:
    private:
        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;
};

#endif /* !RECEPTION_HPP_ */
```

6.51 Reception.hpp

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

#ifndef RECEPTION_HPP_
#define RECEPTION_HPP_

#include <thread>
#include <atomic>
#include <mutex>
#include <unordered_map>


#include "../../common/AException.hpp"
#include "../../common/APizza.hpp"
#include "../../common/Socket.hpp"
#include "../kitchen/Kitchen.hpp"


class Reception {
    public:

    class ErrorReception : public AException {
        public:
            ErrorReception(const std::string &message);
    };


        Reception();
        ~Reception();

        /* Getter */
        int getNbKitchens() const;
        std::vector<std::shared_ptr<Kitchen» getKitchens() const;
        std::shared_ptr<Kitchen> getKitchen(int id) const;

        /* Setter */
        void setValues(int nbCooks, int cookTime, int restockTime,
            bool debug = false);

        /* Methods */
        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);

        /* Reload elem */
        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);

        /* Messefe func handler */
        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);

    protected:
    private:
        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;
        DLLoader<std::shared_ptr<IProcess>(*)()> loader;
};

#endif /* !RECEPTION_HPP_ */
```

6.52 Americana.hpp

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


#include "../../../common/APizza.hpp"

#ifndef AMERICANA_HPP_
#define AMERICANA_HPP_

class AmericanaClass : public APizza {
    public:
        AmericanaClass(int number);
        ~AmericanaClass() override;

        /* Method */
        void cook(int cookTime) override;
        std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient> ingridient)
override;
        void serve() override;

    protected:
    private:
};

#endif /* !AMERICANA_HPP_ */
```

6.53 Americana.hpp

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


#include "../../../common/APizza.hpp"

#ifndef AMERICANA_HPP_
#define AMERICANA_HPP_

class AmericanaClass : public APizza {
    public:
        AmericanaClass(int number);
        ~AmericanaClass() override;

        /* Method */
        void cook(int cookTime) override;
        std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient> ingridient)
override;
        void serve() override;

    protected:
    private:
};

#endif /* !AMERICANA_HPP_ */
```

6.54 Fantasia.hpp

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


#include "../../../common/APizza.hpp"

#ifndef FANTASIA_HPP_
#define FANTASIA_HPP_

class FantasiaClass : public APizza {
    public:
        FantasiaClass(int number);
        ~FantasiaClass() override;

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

    protected:
    private:
};

#endif /* !FANTASIA_HPP_ */
```

6.55 Fantasia.hpp

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


#include "../../../common/APizza.hpp"

#ifndef FANTASIA_HPP_
#define FANTASIA_HPP_

class FantasiaClass : public APizza {
    public:
        FantasiaClass(int number);
        ~FantasiaClass() override;

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

    protected:
    private:
};

#endif /* !FANTASIA_HPP_ */
```

6.56 Margarita.hpp

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

#include "../../../common/APizza.hpp"

#ifndef MARGARITACLASS_HPP_
#define MARGARITACLASS_HPP_

class MargaritaClass : public APizza {
    public:
        MargaritaClass(int number);
        ~MargaritaClass() override;

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


    protected:
    private:
};

#endif /* !MARGARITACLASS_HPP_ */
```

6.57 Margarita.hpp

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

#include "../../../common/APizza.hpp"

#ifndef MARGARITACLASS_HPP_
#define MARGARITACLASS_HPP_

class MargaritaClass : public APizza {
    public:
        MargaritaClass(int number);
        ~MargaritaClass() override;

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


    protected:
    private:
};

#endif /* !MARGARITACLASS_HPP_ */
```

6.58 Regina.hpp

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


#include "../../../common/APizza.hpp"

#ifndef REGINACLASS_HPP_
#define REGINACLASS_HPP_

class ReginaClass : public APizza {
    public:
        ReginaClass(int number);
        ~ReginaClass() override;

        void cook(int cookTime) override;
        std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient> ingridient)
override;
        void serve() override;
    protected:
    private:
};

#endif /* !REGINACLASS_HPP_ */
```

6.59 Regina.hpp

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


#include "../../../common/APizza.hpp"

#ifndef REGINACLASS_HPP_
#define REGINACLASS_HPP_

class ReginaClass : public APizza {
    public:
        ReginaClass(int number);
        ~ReginaClass() override;

        void cook(int cookTime) override;
        std::shared_ptr<Ingridient> prepare(int number, std::shared_ptr<Ingridient> ingridient)
override;
        void serve() override;
    protected:
    private:
};

#endif /* !REGINACLASS_HPP_ */
```

6.60 Utils.hpp

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

#ifndef UTILS_HPP_
#define UTILS_HPP_

class Utils {
    public:
        Utils() = default;
        ~Utils() = default;

        void helper();

    protected:
    private:
};

#endif /* !UTILS_HPP_ */
```

6.61 Utils.hpp

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

#ifndef UTILS_HPP_
#define UTILS_HPP_

class Utils {
    public:
        Utils() = default;
        ~Utils() = default;

        void helper();

    protected:
    private:
};

#endif /* !UTILS_HPP_ */
```