IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

# Simulation d’un écosystème

## Advanced C++ programming

**BERTHOUMIEU Aymeric**
**KINGNE Jéhoiakim**
**RAFISNESQUE Jeremie**
**RIVOAL Samuel**

# Sommaire

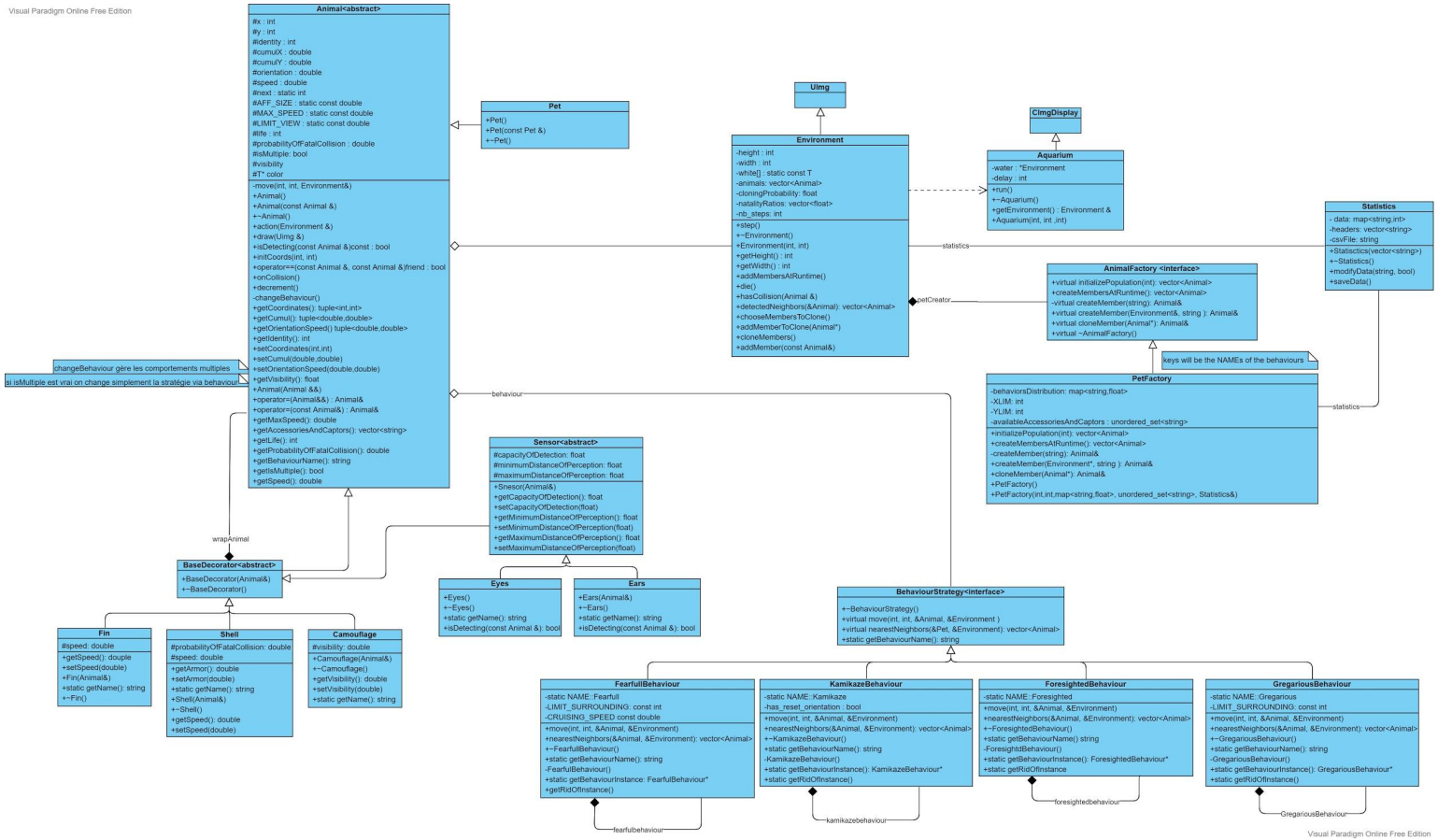


IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

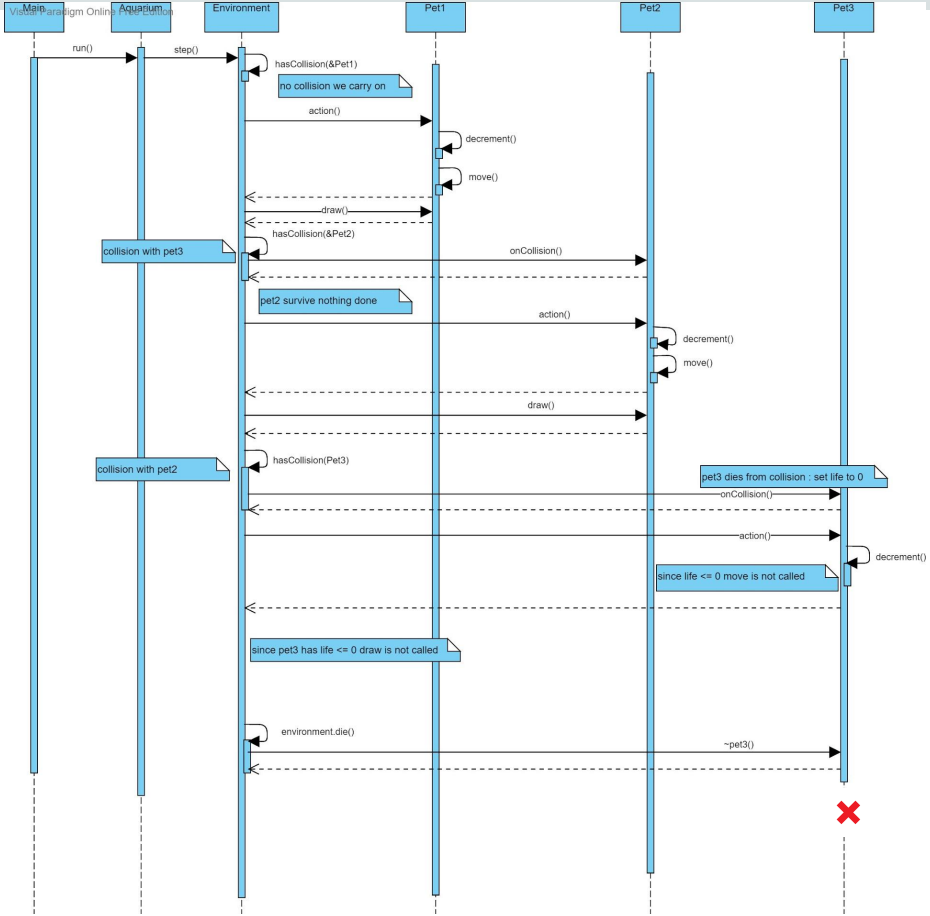
# 1. Modélisation

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

# Modélisation complète

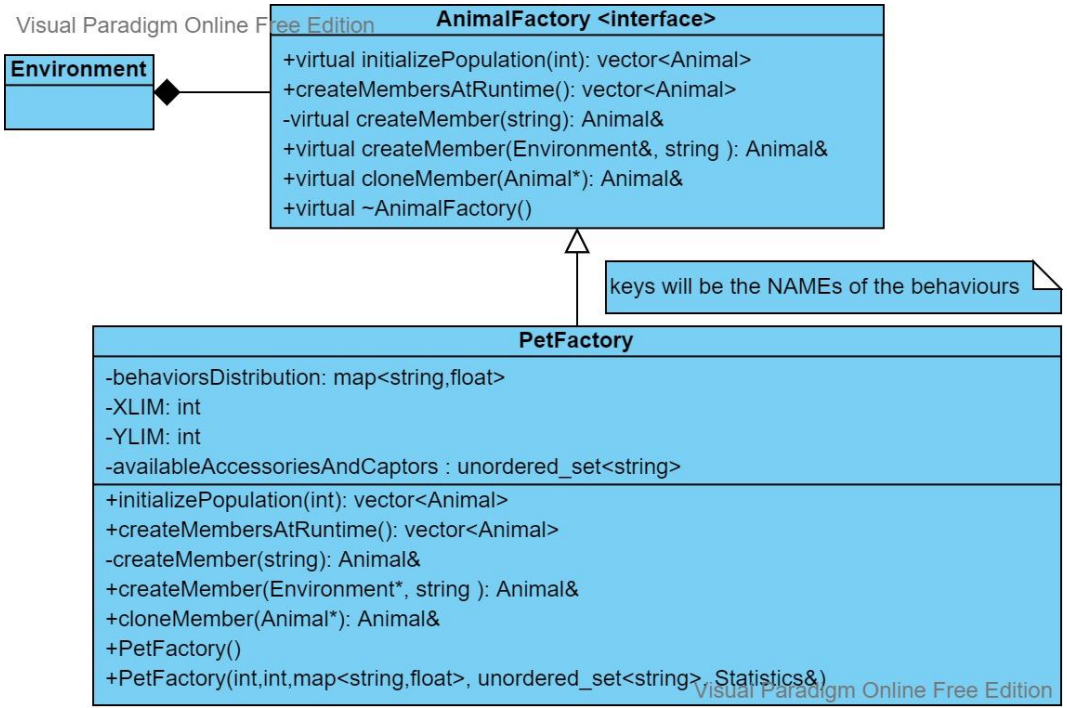


# Exemple : la collision



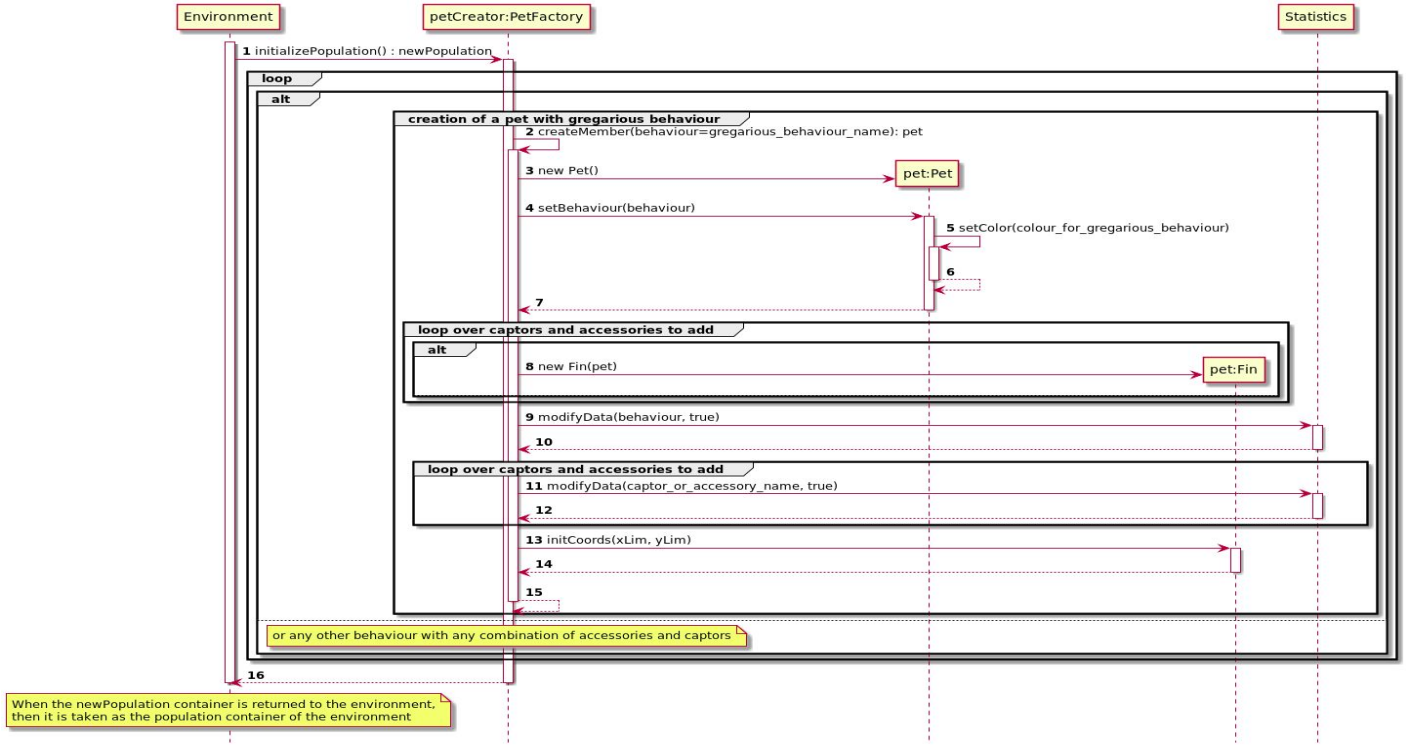
# Design patterns utilisés : Le factory

## Pourquoi ?



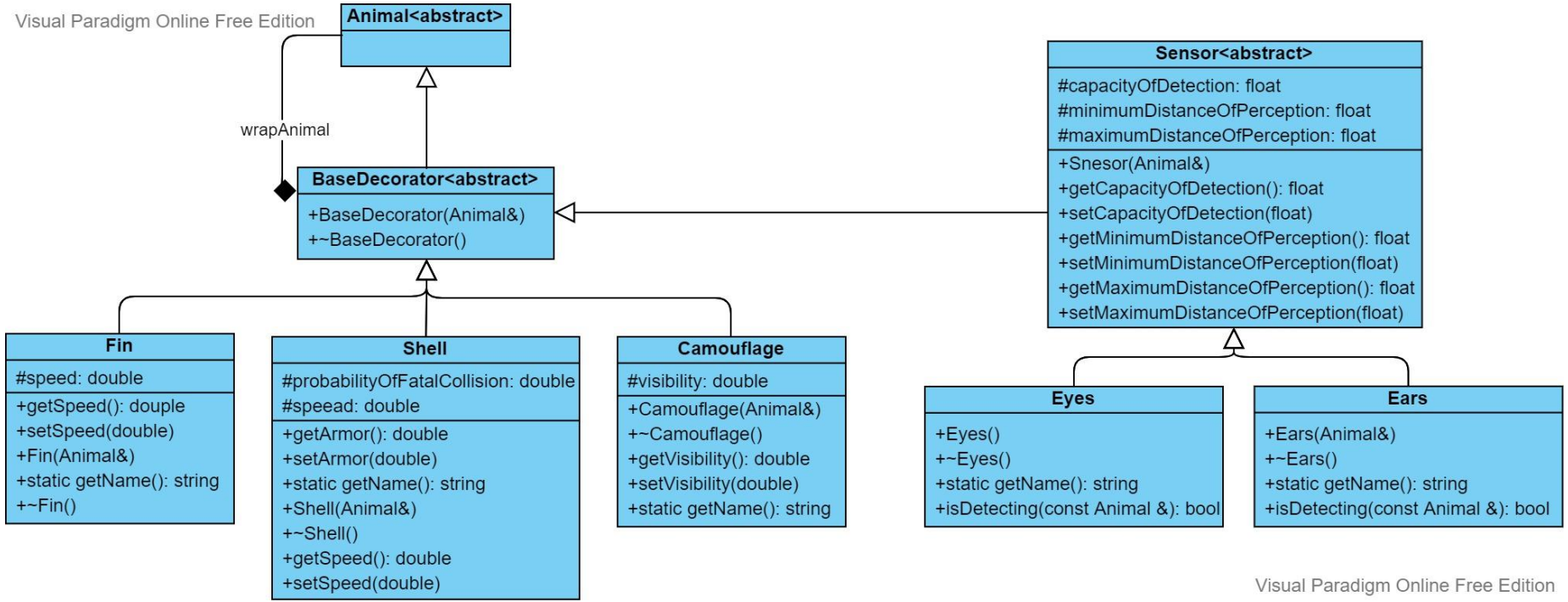
# Design patterns utilisés : Le factory

- une unique porte d'entrée pour créer les bestioles de l'extérieur
- centralisation de la création des bestioles

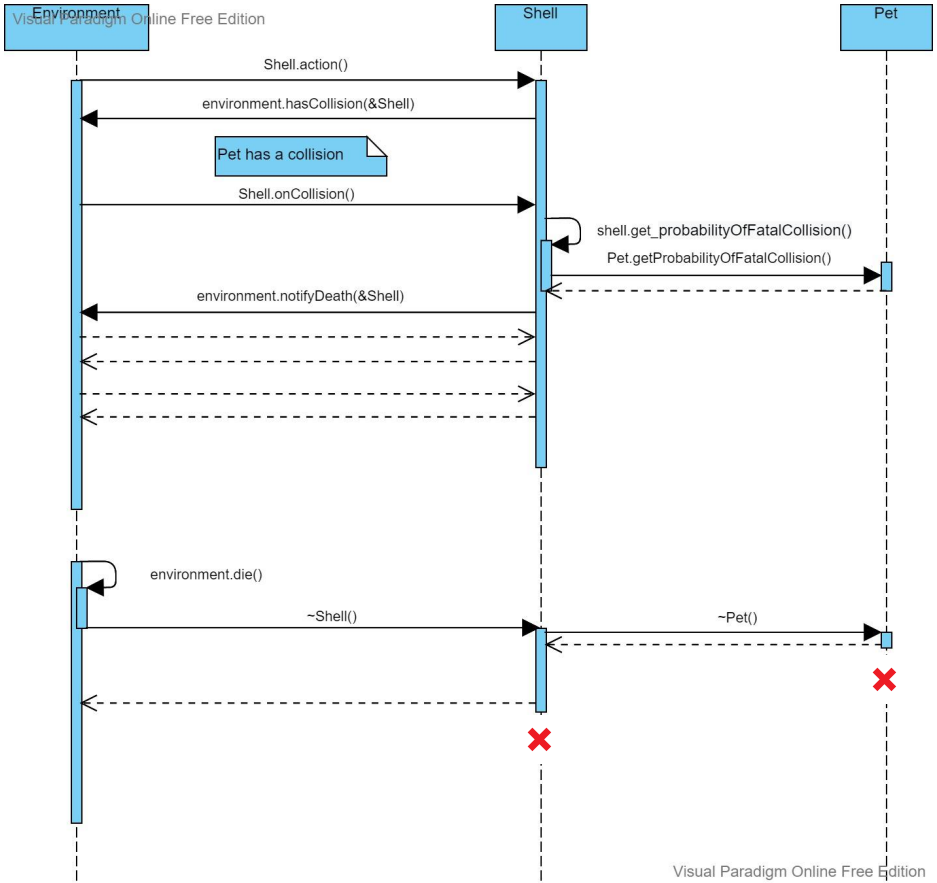


# Design patterns utilisés : Le décorateur

Pourquoi ? - Accumulation d'accessoires et capteurs.
- Destruction et gestion mémoire simples.

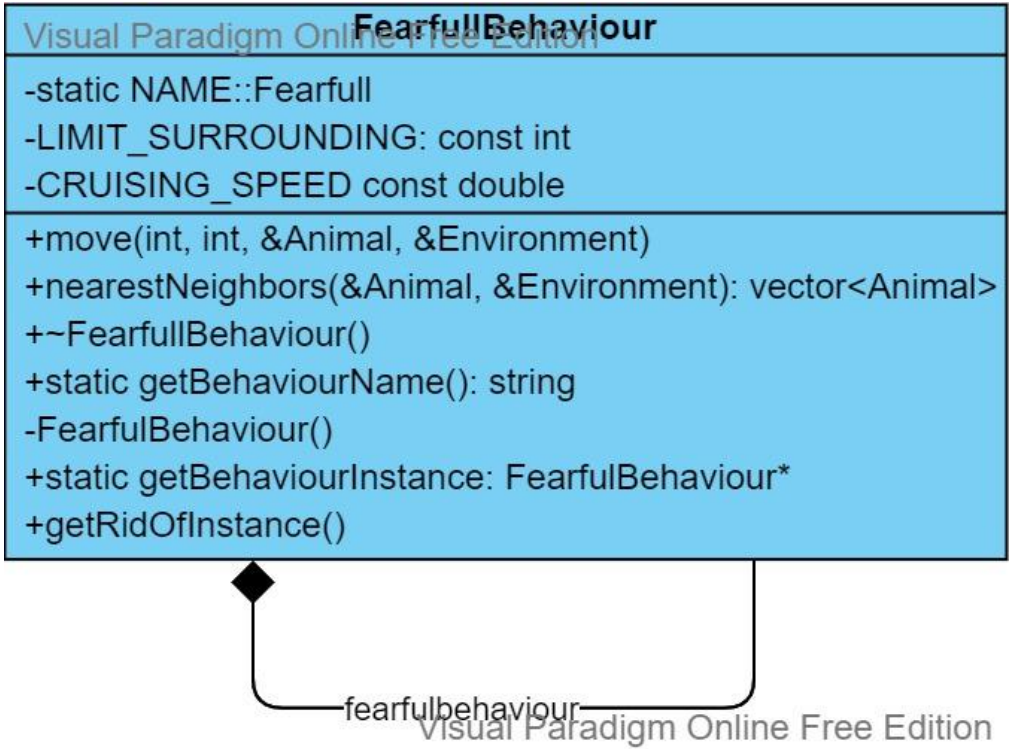


# Design patterns utilisés : Le décorateur



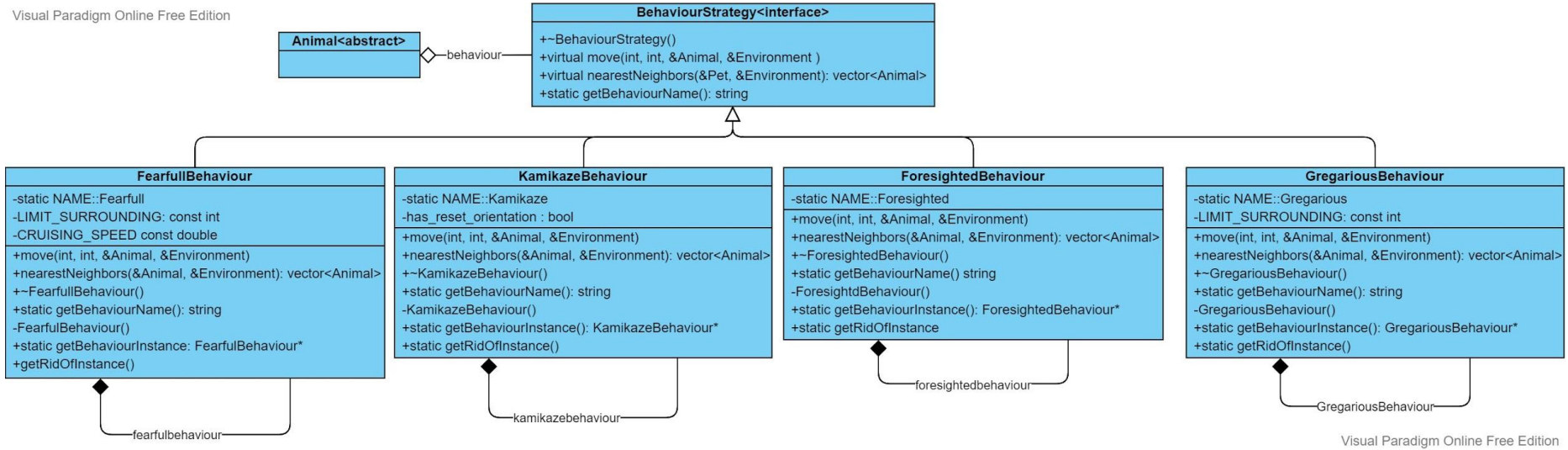
# Design patterns utilisés : Le singleton

Pourquoi ? - Pas de démultiplication inutile d'instances.

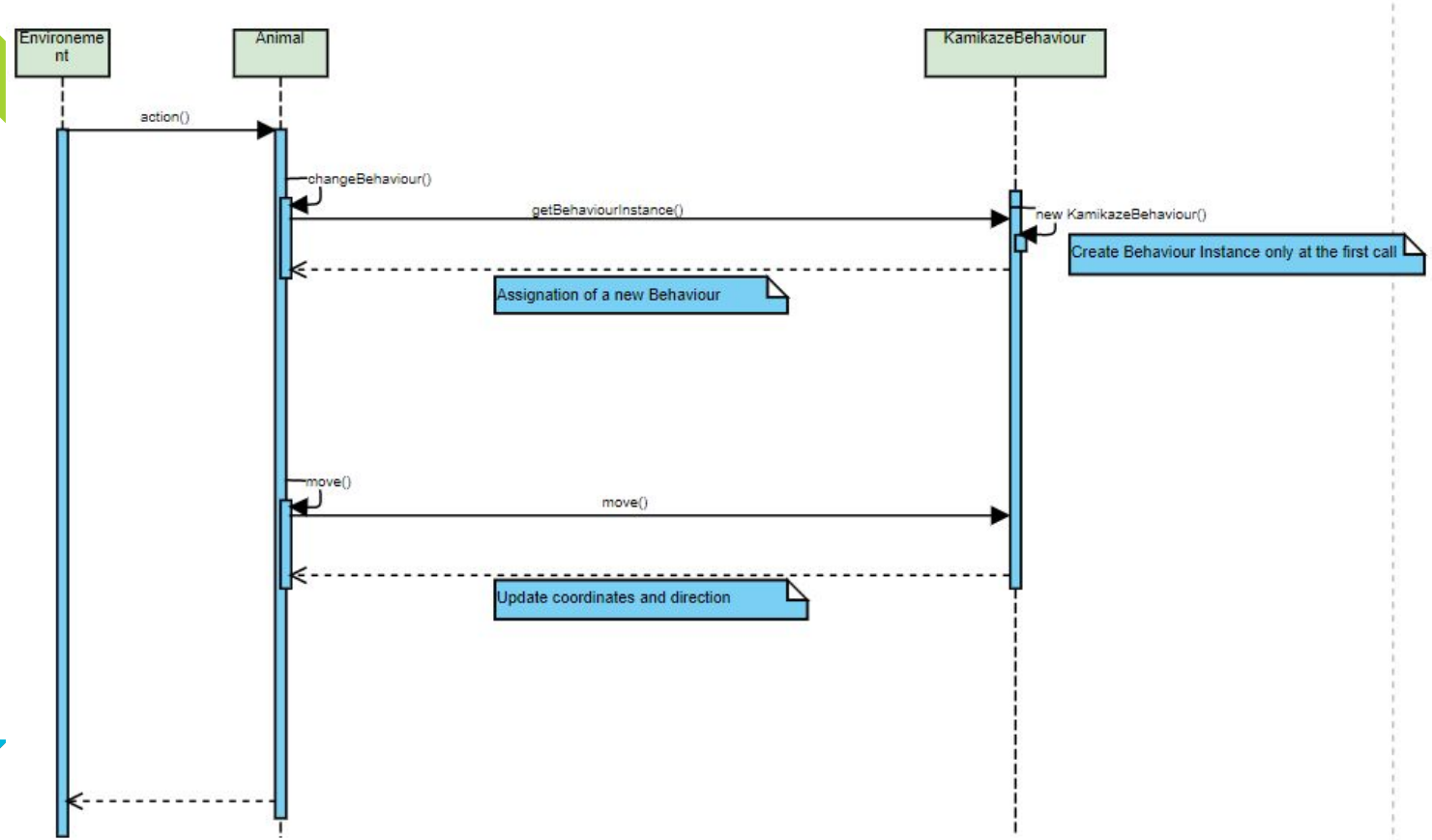


# Design patterns utilisés : Le strategy

Pourquoi ? - Possibilité de changement pendant la simulation.
- Pas de démultiplication d'instance pour faire la même chose grâce au singleton (cf. design pattern précédent).



# Design patterns utilisés : Strategy & Singleton



# 2. Implémentation

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

# Implémentation des comportements

- Interface régissant tous les comportements

```
class BehaviourStrategy
{

public:
    virtual ~BehaviourStrategy(){};
    virtual std::string getBehaviourName() = 0;
    virtual std::vector<Animal *> nearestNeighbors(Animal* pet, Environment& myEnvironment) = 0;
    virtual void move(int xLim, int yLim, Animal* pet, Environment& myEnvironment) = 0;


};
```

# Implémentation des comportements

- Exemple de comportement

```
class GregariousBehaviour: public BehaviourStrategy{
    private:
    static std::string NAME ;
        static const T color[3];

    static GregariousBehaviour* gregariousbehaviour;
    const int LIMIT_SURROUNDING = 1;

    GregariousBehaviour(){};

    public:
    ~GregariousBehaviour();

    static GregariousBehaviour* getBehaviourInstance();
    static void getRidOfInstance();

    std::string getBehaviourName() override;
        static const T* getColor();

    std::vector<Animal *> nearestNeighbors(Animal* pet, Environment& myEnvironment) override;
    void move(int xLim, int yLim, Animal* pet, Environment& myEnvironment) override;

};
```

# Implémentation des comportements

- Instanciation et gestion de vie du comportement

```
GregariousBehaviour* GregariousBehaviour::gregariousbehaviour = nullptr;

void GregariousBehaviour::getRidOfInstance(void){
    delete gregariousbehaviour;
}

GregariousBehaviour* GregariousBehaviour::getBehaviourInstance(){
    if (gregariousbehaviour == nullptr ){
        gregariousbehaviour = new GregariousBehaviour();
    }
    return gregariousbehaviour;
}
```

# Quelques écarts...

- Les accessoires et les capteurs.
- *IsDetecting()*
- Le clonage
- La naissance en cours de simulation

# Tests nominaux

- testDeath1
- testDeath2
- testCollision
- testStatistics
- testBehaviours
- testFearfull
- testGregarious
- testKamikaze
- testMultipleBehaviours

- testShell
- testEars
- testFin
- testPetFactory

# Intégration

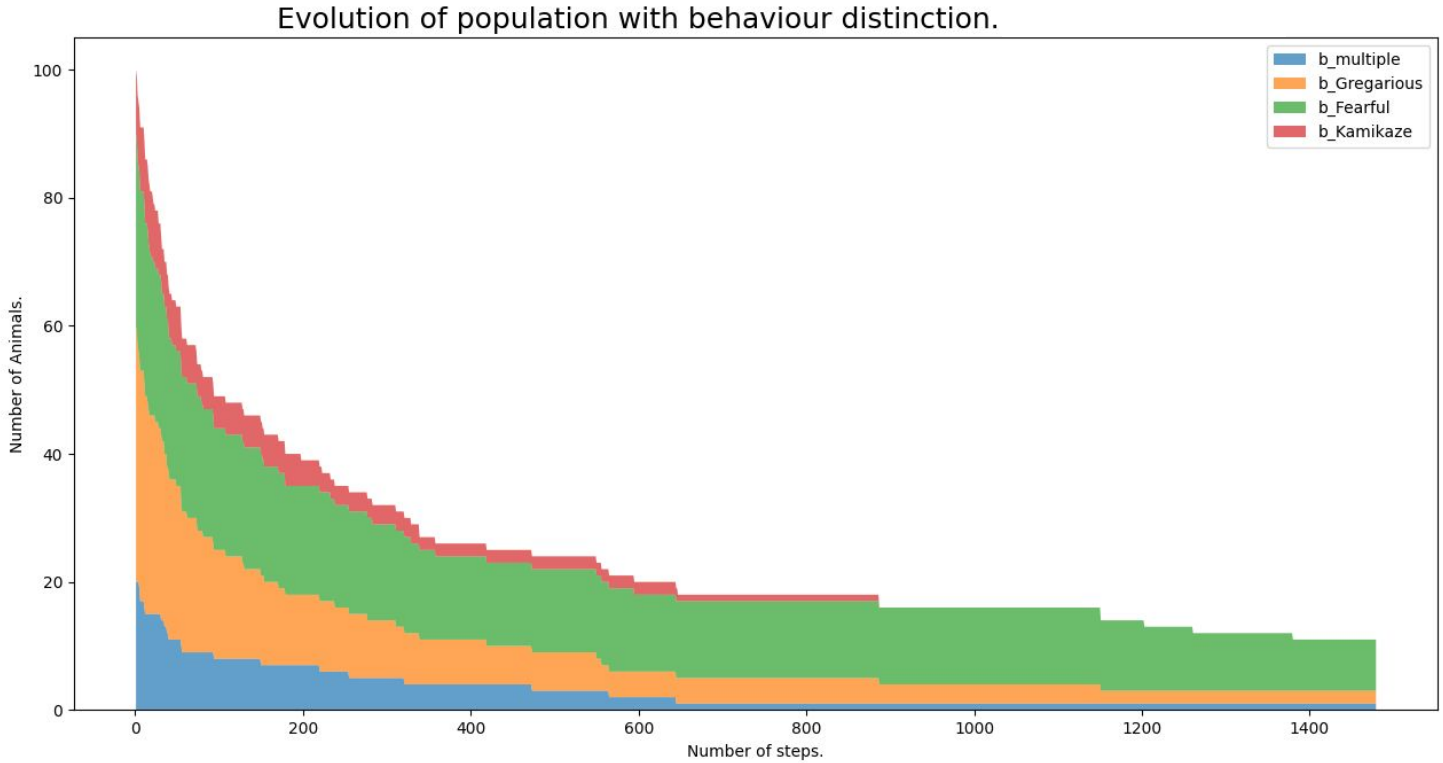
- chacun sa branche sur gitlab
- coder les parties de autres sans les coder entièrement
- merge conflicts

# 3. Quelques résultats

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

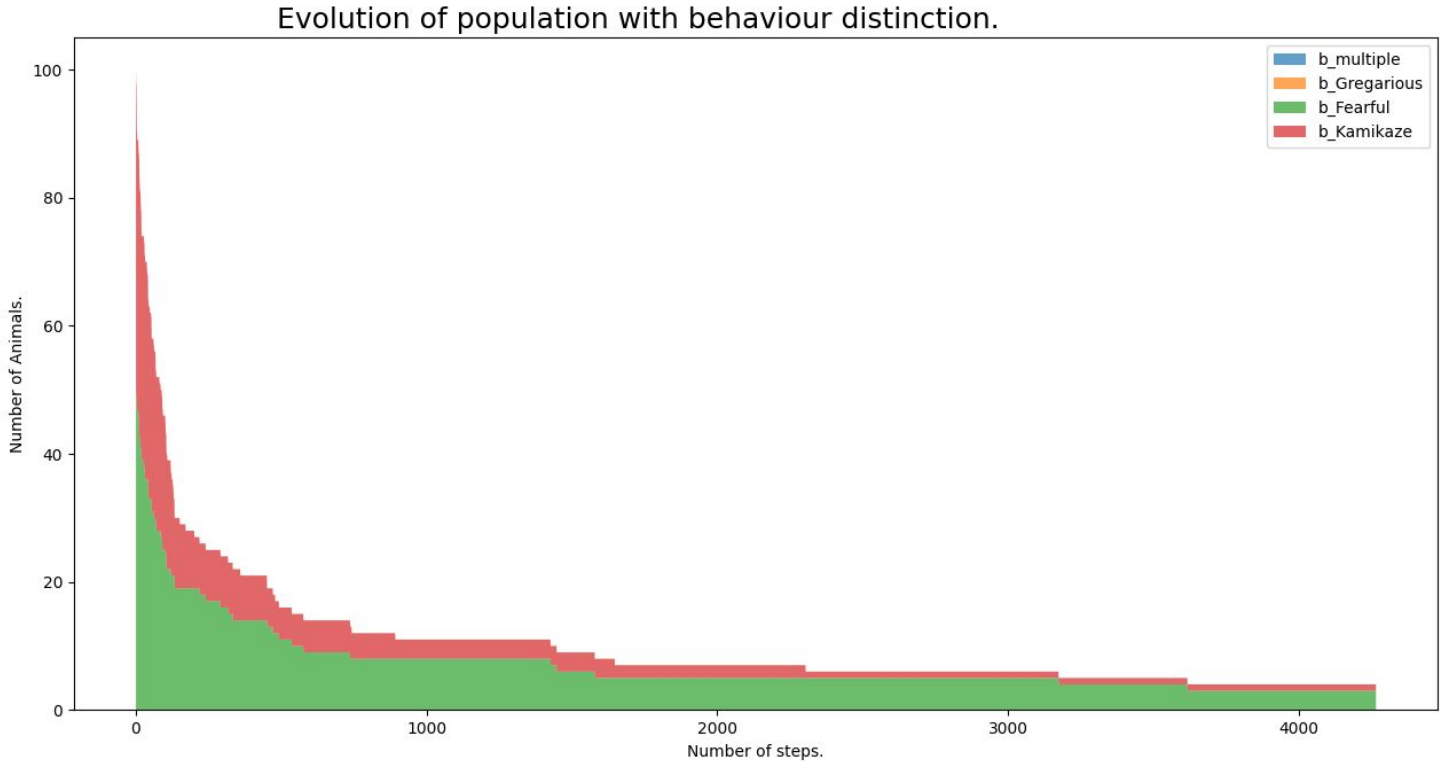
# Simulation 1

100 Pets
Kamikaze 10% ; Fearful 30% ; Gregarious 40% ; Multiple 20%



# Simulation 2

100 Pets
Kamikaze 50% ; Fearful 50% ; Gregarious 0% ; Multiple 0%



# 4. Démonstration

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

# Merci pour votre attention.

Des questions ?