1. 类图

@startuml

class ZooManager {

- name: String

- employeeID: int

- contactInfo: String

+ manageAnimal()

+ manageFacility()

+ manageEmployee()

+ handleFeedback()

}

class Animal {

- name: String

- species: String

- age: int

- quantity: int

+ getInfo()

+ updateInfo()

}

class Facility {

- facilityName: String

- location: String

- status: String

+ maintainFacility()

+ updateStatus()

}

class Employee {

- name: String

- position: String

- hireDate: Date

+ getEmployeeInfo()

+ updateEmployeeInfo()

}

class VisitorFeedback {

- feedbackContent: String

- feedbackTime: Date

- processingStatus: String

+ getFeedback()

+ updateProcessingStatus()

}

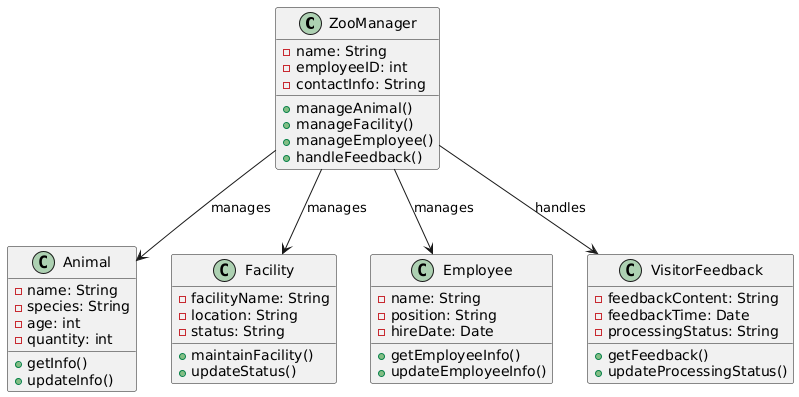
ZooManager --> Animal : manages

ZooManager --> Facility : manages

ZooManager --> Employee : manages

ZooManager --> VisitorFeedback : handles

@enduml



2. 用例图

@startuml

actor ZooManager

usecase "动物信息管理" as UC1

usecase "园区设施管理" as UC2

usecase "员工管理" as UC3

usecase "游客反馈处理" as UC4

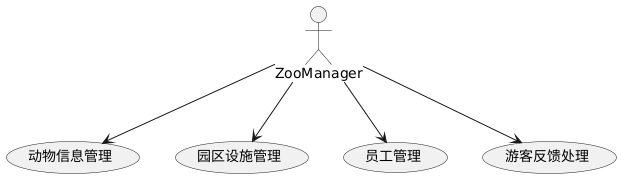
ZooManager --> UC1

ZooManager --> UC2

ZooManager --> UC3

ZooManager --> UC4

@enduml



3. 时序图（添加动物为例）

@startuml

actor ZooManager

participant "动物管理系统" as System

ZooManager -> System: 发起添加动物请求

System -> ZooManager: 提示输入动物信息

ZooManager -> System: 输入动物信息

System -> System: 验证信息合法性

alt 信息不合法

System -> ZooManager: 返回错误提示

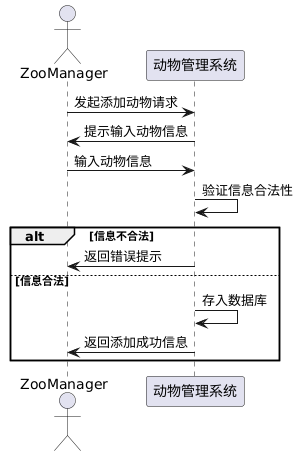
else 信息合法

System -> System: 存入数据库

System -> ZooManager: 返回添加成功信息

end

@enduml



4. 活动图（处理游客反馈为例）

@startuml

start

:游客反馈到达;

:管理人员接收反馈;

:判断反馈类型;

fork

:投诉;

:分配给相关部门处理;

:等待处理结果;

:更新处理状态;

fork again

:建议;

:评估可行性;

if (可行?) then (yes)

:实施并记录;

else (no)

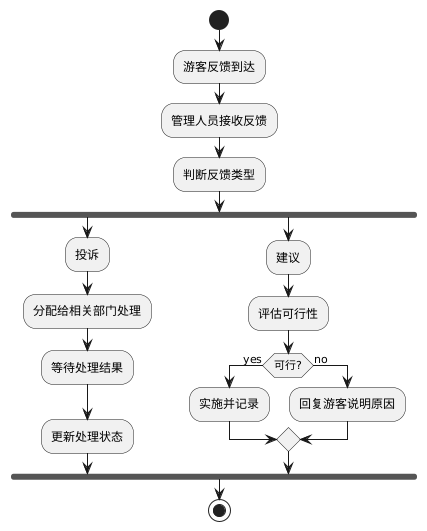
:回复游客说明原因;

endif

end fork

stop

@enduml



5. 状态图（动物状态为例）

@startuml

[\*] --> 新入园

新入园 --> 健康状态

健康状态 --> 生病状态 : 生病

生病状态 --> 健康状态 : 康复

生病状态 --> 死亡状态 : 死亡

死亡状态 --> [\*]

@enduml

