@startuml

|Пользователь|

start

:Приложение открыто;

|Логика|

:Вывод окна размещения;

|Пользователь|

:Выбран пункт меню "Открыть конфигурацию";

:Выбран файл;

|Логика|

while (соответствующий тип ?) is (нет)

:Вывод сообщения "Неверный тип файла";

endwhile (да)

:Загрузка конфигурации в таблицу;

|Пользователь|

:Формирование плана выращивания;

|Логика|

while (есть ли ошибки ?) is (да)

:Вывод сообщения "Ошибки в плане выращивания!";

endwhile (нет)

:Приведение плана к нормальному виду;

|Пользователь|

:Нажата кнопка "сохранить";

|Логика|

:формирование XML-файлов;

|База данных|

:Сохранение конфигурации;

:Сохранение плана выращивания;

|Пользователь|

:Нажата кнопка "запустить";

|Логика|

:создание объекта типа "диспетчер";

stop

@enduml

@startuml

Dispatcher ..> IGrowingPlanCommon

IGPAllowedStates <.. IGrowingPlanCommon

IInstruction <.right. IGPAllowedStates

GrowingPlanCommon --|> IGrowingPlanCommon

GrowingPlanCommon .. IGPAllowedStates

StateFormersController --|> IStateFormersController

StateFormersController --\* Dispatcher

class GrowingPlanCommon{

IList<IGPAllowedStates> AllowedStatesList

}

interface IStateFormersController{

IList<IMeasurment> FormDevicesInstructions()

}

class StateFormersController{

-StateFormers

}

class Dispatcher {

-devicesController;

-sensorsController;

-growingPlan;

void RunGrowing()

void StopGrowing()

void Tic\_Toc()

Boolean AffectEnvironmentByStates()

}

interface IInstruction{

IMeasurment GetMaxAllowedState()

IMeasurment GetMinAllowedState()

}

interface IGrowingPlanCommon{

IGPAllowedStates GetAllowedStates()

String CheckGrowingPlan()

}

interface IGPAllowedStates{

String Name

Int32 Hours

Int32 Minutes

Double Progress

IInstruction GetStateByPropertyID

}

@enduml

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

@startuml

class Nature{

-temperatureReal

-phReal

-lightReal

Thread EnvironmentThread()

void Change()

}

IDevice <|-- Device

Device <|-- TemperatureDevice

Device <|-- LightDevice

Device <|-- PhDevice

TemperatureDevice --o DevicesController

LightDevice --o DevicesController

PhDevice --o DevicesController

IDispatcher <|-- Dispatcher

IDispatcher .left. IGPAllowedStates

Dispatcher \*-- DevicesController

Dispatcher \*-- SensorsController

Dispatcher \*-right- ActiveSensorsController

ISensor <|-- Sensor

Sensor <|-- TemperatureSensor

Sensor <|-- LightSensor

Sensor <|-- PhSensor

TemperatureSensor --o SensorsController

LightSensor --o SensorsController

PhSensor --o SensorsController

class ActiveSensorsController{

Thread MonitorSensorsThread

void StopMonitoring()

void RunMonitoring()

void Monitor()

}

interface IDispatcher {

void RunGrowing()

void StopGrowing()

DateTime GetCurrentTime()

}

class Dispatcher {

-devicesController;

-sensorsController;

-growingPlan;

void RunGrowing()

void StopGrowing()

void Tic\_Toc()

Boolean AffectEnvironmentByStates()

}

interface IGPAllowedStates{

String Name

Int32 Hours

Int32 Minutes

Double Progress

IInstruction GetStateByPropertyID

}

class DevicesController{

#physicalObjectsController

void AffectEnvironment()

}

class SensorsController{

#physicalObjectsController

IList<IMeasurment> GetEnvironmentStates()

}

interface IDevice{

void SetState();

IMeasurment GetState();

}

interface ISensor{

IMeasurment Measure();

IMeasurment GetLastMeasurment();

}

class Device{

#State

#location

+Name

void SetState()

IMeasurment GetState()

}

class Sensor{

#lastMeasurment

#location

Location GetLocation()

void SetLocation()

IMeasurment Measure()

}

@enduml