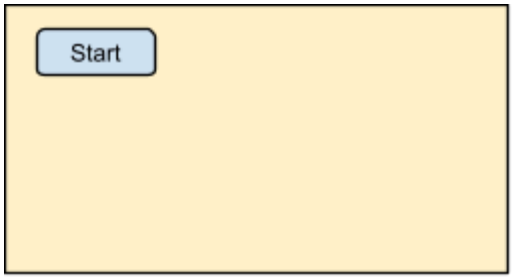
StateMachine - абстрактен, теоритичен

WorkflowEngine, реализиран със машина на състоянията StateMachine.

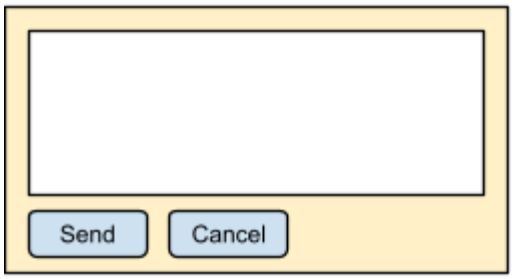
Да се напише библиотека за управление на потока на работа (workflow).

Определение:
Машината на състоянията се състои от множество състояния, множество събития и функция на преходите, и текущо състояние.
Функцията на преходите е <състояние, събитие> -> състояние. Тя определя, в кое е следващото състояние на StateMachine-ата, при дадено събитие. Т.е. <текущо състояние, събитие> -> следващо състояние. Тази функция може да не е напълно дефинирана, т.е. да има двойки <състояние, събитие> за които да няма резултат. В такива случаи, машината не прави преход.
При промяна на състоянието се случват следните неща:
- изпълняват се OnAfterInvokation неща за текущото състояние
- изпълняват се OnBeforeInvokation неща за новото състояние и то става текущо.

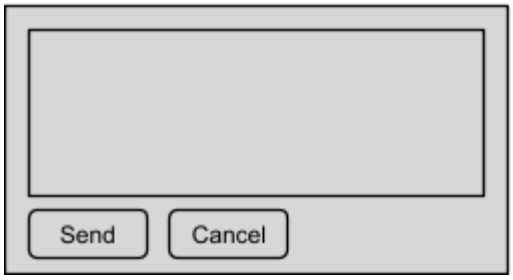
Пример. Форма за изпращане на имейли. Тя има три състояния.



Начално състояние
- като се натисне Start, се отива във долното



Потребителят пише
- Send води до долното състояние
- Cancel - до горното



Съобщението се праща (това отнема някакво време)
- тук контролите са посивени, защото са неактивни
- когато съобщението се прати, се връща в най-горното

```
//състояния
enum {
  Initial,
  WritingEmail,
  SendingEmail
} FormStates;

//съобщения
enum {
  StartWriting,
  SendEmail,
  EmailSent,
  CancelEmail
} FormEvents;

....
//в конструктора на формата (или там където трябва).
this->startButton = new Button();
this->sendButton = new Button();
this->messageArea = new TextArea();
this->cancelButton = new Button();

this->stateMachine = new StateMachine<FormStates, FormEvents>(Initial);

//сега се конфигурира машинарията
this->stateMachine->addTransition(Initial, StartWriting, WritingEmail);
this->stateMachine->addTransition(WritingEmail, SendEmail, SendingEmail);
this->stateMachine->addTransition(SendingEmail, EmailSent, Initial);
//това събитие ще се праща от "вътре"
this->stateMachine->addTransition(WritingEmail, CancelEmail, Initial);
//на Cancel бутона

this->initialStatePreEntry = new BeforeInitialStateInvokation(
    this->startButton, this->sendButton, this->cancelButton, this-> messageArea);
this->stateMachine->addBeforeStateInvokation(Initial, this->initialStatePreEntry);

this->initialStatePostExit = new AfterInitialStateInvokation(
    this->startButton, this->sendButton, this->cancelButton, this->messageArea);
this->stateMachine->addAfterStateInvokation(Initial, this->initialStatePostExit);
//... добавяме и за останалите състояния
// BeforeInitialStateInvokation и AfterInitialStateInvokation са класове, които
//"държат" поведението за "влизане" и "излизане" в и от началното състояние на
//машината

//още сме в нашата формата
void start() //се вика като се цъкне startButton-а
```

```
{
  this->stateMachine->sendEvent(StartWriting);
}
void send() //вика се от sendButton-а
{
  this->stateMachine->sendEvent(SendEmail);
}
void emailSent()//частен метод, вика се от формата след като имейлът е изпратен
{
  this-stateMachine->sendEvent(EmailSent);
}

в BeforeInitialStateInvokation класа
//...
this->startButton->show();
this->sendButton->hide();
this->messageArea->hide();
this->cancelButton->hide();
```

Каква е схемата всъщност?
Когато нещо се случи, ние пращаме подходящото събитие на вътрешната stateMachine. Тя проверява, дали за текущото състояние е дефиниран преход за това събитие. Изпълнява OnAfterInvokation нещото за текущото състояние и OnBeforeInvokation за идното състояние.

За едно състояние и едно събитие може да има само едно състояние. Т.е. второто добавяне на преход от дадено състояние със събитие замества първото.

За едно събитие може да има по повече от едно OnBeforeInvokation и OnAfterInvokation. Изпълняват се в реда на добавяне.

ЗАДАЧА!
Да се напише библиотечката и да се демонстрира с приложна област. Може и с дадения пример. Ще трябва да си напишете добре бутоните и т.н.