--------------------- Teller von Chabir ---------------------

---------------------- Teller.cpp ----------------------------

```
 /* Created on 22. Oktober 2019, 11:12
 */
#include <iostream>
#include "Teller.h"
#include <cstdlib>
using namespace std;

Teller::Teller() {
    pancakeNum = 0;

}

Teller::Teller(int max) {
    maxPancake = max;
     pancakeNum = 0;
}

Teller::~Teller() {
}



void Teller::put(void){
    if (pancakeNum == 10){
        cout << " \n Teller ist VOLL !" << endl;
    }else{
        pancakeNum += 1;
    }
}
void Teller::remove(void){
    if (pancakeNum == 0){
        cout << " \n Teller ist LEER !" << endl;
    }else{
        pancakeNum -= 1;
    }
}

int Teller::getPanckakeNum(void){
    printf("\n Pfannkuchen auf dem Teller: %d", pancakeNum);
    return pancakeNum;
}
```

---------------------------------- Teller.h ----------------------

```
#include <cstdlib>
#include <iostream>
```

```
#ifndef TELLER_H
#define TELLER_H

class Teller {
public:
    Teller();
    Teller(int);
//    Teller(const Teller& orig);
    virtual ~Teller();
    /*FCTS*/
    void put(); // adds 1
    void remove(); // removes 1
    int getPanckakeNum(void);
private:
    int pancakeNum  ;
    int maxPancake ;
};

#endif /* TELLER_H */

*------------------FSMTeller.cpp --------------------*

#include "FSMTeller.h"
#include <iostream>
#include <stdlib.h>

#define SIFR 0
#define ZEHN 10



using namespace std;



FSMTeller::FSMTeller() {
    cout << "\n bin da" << endl;
    currentState = LEER;
}

FSMTeller::FSMTeller(Teller theTeller) {
    this->theTeller = theTeller;

}



FSMTeller::~FSMTeller() {
}

void FSMTeller::evalStates(string command){

    int now = 0;
    printf("\n Aktueller Zustand ist : %d", currentState);
```

```
    switch(currentState){
        case LEER:
            if(command == "put"){
                theTeller.put();
                theTeller.getPanckakeNum();
                currentState = GEFUELLT;
            }


            if (command == "remove"){
                theTeller.remove();
                theTeller.getPanckakeNum();
            }

            if(command != "put" && command != "remove"){
                printf("\n Lernen Sie Schreiben!!");
            }

            break;
        case GEFUELLT:
            if(command == "put"){
                theTeller.put();
                now = theTeller.getPanckakeNum();
                if(now == ZEHN){
                    currentState = VOLL;
                }
            }

            if (command == "remove"){
                theTeller.remove();
                theTeller.getPanckakeNum();
                printf("\n Aktueller Zustand ist : %d", currentState);
                if(theTeller.getPanckakeNum() == SIFR){
                    printf("\n Aktueller Zustand ist : %d",
currentState);
                    currentState = LEER;
                }
            }

            if(command != "put" && command != "remove"){
                printf("\n Lernen Sie Schreiben!!");
            }

            break;

        case VOLL:
            printf("\n Aktueller Zustand ist : %d", currentState);
            if(command == "put"){
                theTeller.put();
                theTeller.getPanckakeNum();
            }
            if(command == "remove"){
```

```
                theTeller.remove();
                theTeller.getPanckakeNum();
                if(theTeller.getPanckakeNum() < ZEHN) {
                    currentState = GEFUELLT;
                    printf("\n Aktueller Zustand ist : %d", LEER);
                }
            }

            if(command != "put" && command != "remove"){
                printf("\n Lernen Sie Schreiben!!");
            }

            break;

            }

     printf("\n Neuer Zustand ist : %d", currentState);



}

int FSMTeller::getState() {
    return (int) currentState;
}


*---------------------------- FSMTeller.h --------------------*



#include "Teller.h"

using namespace std;

#ifndef FSMTELLER_H
#define FSMTELLER_H
enum STATES {
    LEER,
    GEFUELLT,
    VOLL,
};

class FSMTeller {
public:
    FSMTeller();
    FSMTeller(Teller);
    virtual ~FSMTeller();
    void evalStates(string);
    int getState();


private:
    STATES currentState;
```

```
    Teller theTeller;

};

#endif /* FSMTELLER_H */

*---------------------main von Chabir's Teller ------------------*

#include <cstdlib>
#include <iostream>

#include "Teller.h"
#include "FSMTeller.h"

using namespace std;

/*
 *
 */
int main(int argc, char** argv) {

//    //Manuelles Testen
    Teller t(10);
//    t.put();
//     t.getPanckakeNum();

//
//    for (int i = 0; i < 20; i++) {
//        t.put();
//    }
//    t.getPanckakeNum();



        FSMTeller myFSM;

        string command;

           while (true) {          // Immerwieder nachfragen, was Baby
 machen soll.

 cout << "\n Geben sie put oder remove ein: " << endl;
                cin >> command;
             myFSM.evalStates(command);   // Evaluierungslogik
 der FSM in der Schleife aufrufen.
           }

    return 0;
}

*------------------------------ Test von Chabir's Teller
 --------------------*
```

```
#include "gtest/gtest.h"
#include "FSMTeller.h"
#define NEUN 9

TEST(Test, test){

    FSMTeller t;
    EXPECT_EQ(t.getState(),0);      //Zustand Leer

    t.evalStates("put");
    EXPECT_EQ(t.getState(),1);      //Zustand Gefüllt

    for(int i = 1; i<=NEUN; i++){
        t.evalStates("put");
    }
    EXPECT_EQ(t.getState(),2);      //Zustand Voll

}

TEST(Test, test1){

    Teller t1(3);
    t1.put();
    EXPECT_EQ(t1.getPanckakeNum(),1);
    t1.put();
    EXPECT_EQ(t1.getPanckakeNum(),2);

}
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
Error using dbstatus
Error: File: C:\PR2\ChabirsTeller.m Line: 1 Column: 1
Invalid use of operator.
```

Published with MATLAB® R2018b