```
/*
 * File:   UBahnFSM.h
 * Author: root
 *
 * Created on 27. Januar 2020, 12:30
 */

#ifndef UBAHNFSM_H
#define UBAHNFSM_H

enum UBahnState{DRIVING, STOPPED, ERROR, CLEAN_UP};
enum UBahnEvent{START_DRIVING, STOP, PASSENGER_ENTER,
 PASSENGER_LEAVE,CLEAN_UP_DONE};

class UBahnFSM {
public:
    UBahnFSM();
    virtual ~UBahnFSM();

    void evalEvents(UBahnEvent event);
    void evalState();
    UBahnState getState();

private:
    UBahnState state;
    UBahnEvent event;
};

#endif /* UBAHNFSM_H */
```

```
/*
 * File:   UBahnFSM.cpp
 * Author: root
 *
 * Created on 27. Januar 2020, 12:30
 */

#include "UBahnFSM.h"
using namespace std;
#include <iostream>

UBahnFSM::UBahnFSM() {

    state = DRIVING;        //Startpunkt für die Tests
}

UBahnFSM::~UBahnFSM() {
}
```

```
void UBahnFSM::evalEvents(UBahnEvent event) {

    switch (state) {
        case DRIVING:
            if (event == PASSENGER_LEAVE) {

 cout << "Passagier verlässt während der Fahrt den Zug" << endl;
                state = ERROR;
            } else if (event == PASSENGER_ENTER) {
                cout << "Passagier betritt den Zug" << endl;
                state = CLEAN_UP;
            } else if (event == STOP) {
                cout << "Zug stoppt" << endl;
                state = STOPPED;
            }
            break;

        case STOPPED:
            if (event == PASSENGER_LEAVE) {

 cout << "Passagier verlässt den haltenden Zug" << endl;
                state = STOPPED;
            } else if (event == PASSENGER_ENTER) {
                cout << "Passagier betritt den haltenden Zug" << endl;
                state = STOPPED;
            } else if (event == START_DRIVING) {
                cout << "Zug fährt an" << endl;
                state = DRIVING;
            }
            break;

        case CLEAN_UP:
            if (event == CLEAN_UP_DONE){
             cout << "ist sauber" << endl;
             state = STOPPED;
            }
            break;

        case ERROR:
            break;
        default:
             cout << "Irgendwas ist schief gelaufen" << endl;
            break;
    }
}

void UBahnFSM::evalState() {

}

UBahnState UBahnFSM::getState() {
    return state;
}
```

```
/*
 * File:   UBahn.h
 * Author: root
 *
 * Created on 27. Januar 2020, 10:50
 */

#ifndef UBAHN_H
#define UBAHN_H

#include<time.h>
#include "IZug.h"

class UBahn : public IZug{      //nichtb das public vergessen
public:
    UBahn(int);
    virtual ~UBahn();
    void drive();   //printed"U(line)driving"
    void stop();    //printed"U(line)stopping"
    void arrive();  //printf "U(line)arrived at (current time)"
    void enter() override;  //Funktionen aus dem Interface,
 welche hier eingefügt werden müssen
    void leave() override;  //Funktionen aus dem Interface,
 welche hier eingefügt werden müssen

private:
    int line;       //Ob U1,U2....
    struct tm *theTime; //struct
 damit die Uhrzeit verwendet werden kann
    time_t peter;   //time ist ein Attribut vom datentyp time_t

};

#endif /* UBAHN_H */

/*
 * File:   UBahn.cpp
 * Author: root
 *
 * Created on 27. Januar 2020, 10:50
 */

#include "UBahn.h"
#include <time.h>
#include <iostream>
#include <cstdlib>
#include <ios>
#include <iomanip>

using namespace std;


UBahn::UBahn(int line) {
    this->line = line;
}

UBahn::~UBahn() {
}

void UBahn::drive() {
    cout<<"Die U"<<line<<"driving"<<endl;
}

void UBahn::stop() {
    cout<<"Die U"<<line<<"is stopping"<<endl;
}

void UBahn::arrive() {

    peter = time(NULL);
    theTime=localtime(&peter);
    cout<<"Die U"<<line<<" arrived at "<<theTime->tm_mday<<"/"
            <<theTime->tm_mon+1<<"/"<<theTime->tm_year-100<<"\n"
            <<setw(20)<<right<<theTime->tm_hour<<":"<<theTime-
>tm_min<<":"
            <<theTime->tm_sec<<endl;
}

void UBahn::enter() {
    cout<<"ich steige ein"<<endl;
}

void UBahn::leave() {
cout<<"ich hau ab"<<endl;
}



/*
 * File:   Person.h
 * Author: root
 *
 * Created on 27. Januar 2020, 11:50
 */

#ifndef PERSON_H
#define PERSON_H
#include "IZug.h"

class Person {
public:
    Person();
    virtual ~Person();
    void enterTrain(IZug *zug);
    void leaveTrain();

private:
    IZug *zug;

};

#endif /* PERSON_H */


/*
 * To change this license header,
 choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * File:   Person.cpp
 * Author: root
 *
 * Created on 27. Januar 2020, 11:50
 */

#include "Person.h"

Person::Person() {
}

Person::~Person() {
}

void Person::enterTrain(IZug* zug) {
    this->zug = zug;
    zug ->enter();
}

void Person::leaveTrain() {
    zug->leave();
}



/*
 * File:   IZug.h
 * Author: root
 *
 * Created on 27. Januar 2020, 11:45
 */

#ifndef IZUG_H
#define IZUG_H

enum ZugTyp {PUBLC, TRANSPORT};

class IZug{
    public:
        virtual void enter() = 0;
        virtual void leave() = 0;
protected:
    int passenger_count;
    ZugTyp type;

};

#endif /* IZUG_H */
```

```
/*
 * File:   main.cpp
 * Author: root
 *
 * Created on 27. Januar 2020, 10:48
 */

#include <cstdlib>

#include "UBahn.h"
#include "Person.h"
#include "IZug.h"


using namespace std;

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

    UBahn bahn(2);
    bahn.arrive();

    Person Passagier;
    Passagier.enterTrain(&bahn);
    Passagier.leaveTrain();


    return 0;
}
```

```
/*
 * File:   newsimpletest1.cpp
 * Author: root
 *
 * Created on 27. Januar 2020, 13:22
 */

#include "UBahnFSM.h"
#include "gtest/gtest.h"

//enum UBahnEvent{START_DRIVING, STOP, PASSENGER_ENTER,
 PASSENGER_LEAVE,CLEAN_UP_DONE};
TEST(T1, Zustände){

    UBahnFSM Peter;
    Peter.evalEvents(PASSENGER_ENTER);
    Peter.evalEvents(CLEAN_UP_DONE);
    Peter.evalEvents(START_DRIVING);
    Peter.evalEvents(PASSENGER_LEAVE);

    EXPECT_EQ(Peter.getState(),ERROR);

}

TEST(T2, Transitionen){

    UBahnFSM Zwicker;
    Zwicker.evalEvents(PASSENGER_ENTER);
    Zwicker.evalEvents(CLEAN_UP_DONE);
    Zwicker.evalEvents(PASSENGER_ENTER);
    Zwicker.evalEvents(PASSENGER_LEAVE);
    Zwicker.evalEvents(START_DRIVING);
    Zwicker.evalEvents(STOP);
    Zwicker.evalEvents(START_DRIVING);
    Zwicker.evalEvents(PASSENGER_LEAVE);

    EXPECT_EQ(Zwicker.getState(),ERROR);
}
```

```
Error using dbstatus
Error: File: C:\PR2\UbahnMarc.m Line: 2 Column: 1
Invalid use of operator.
```

*Published with MATLAB® R2018b*