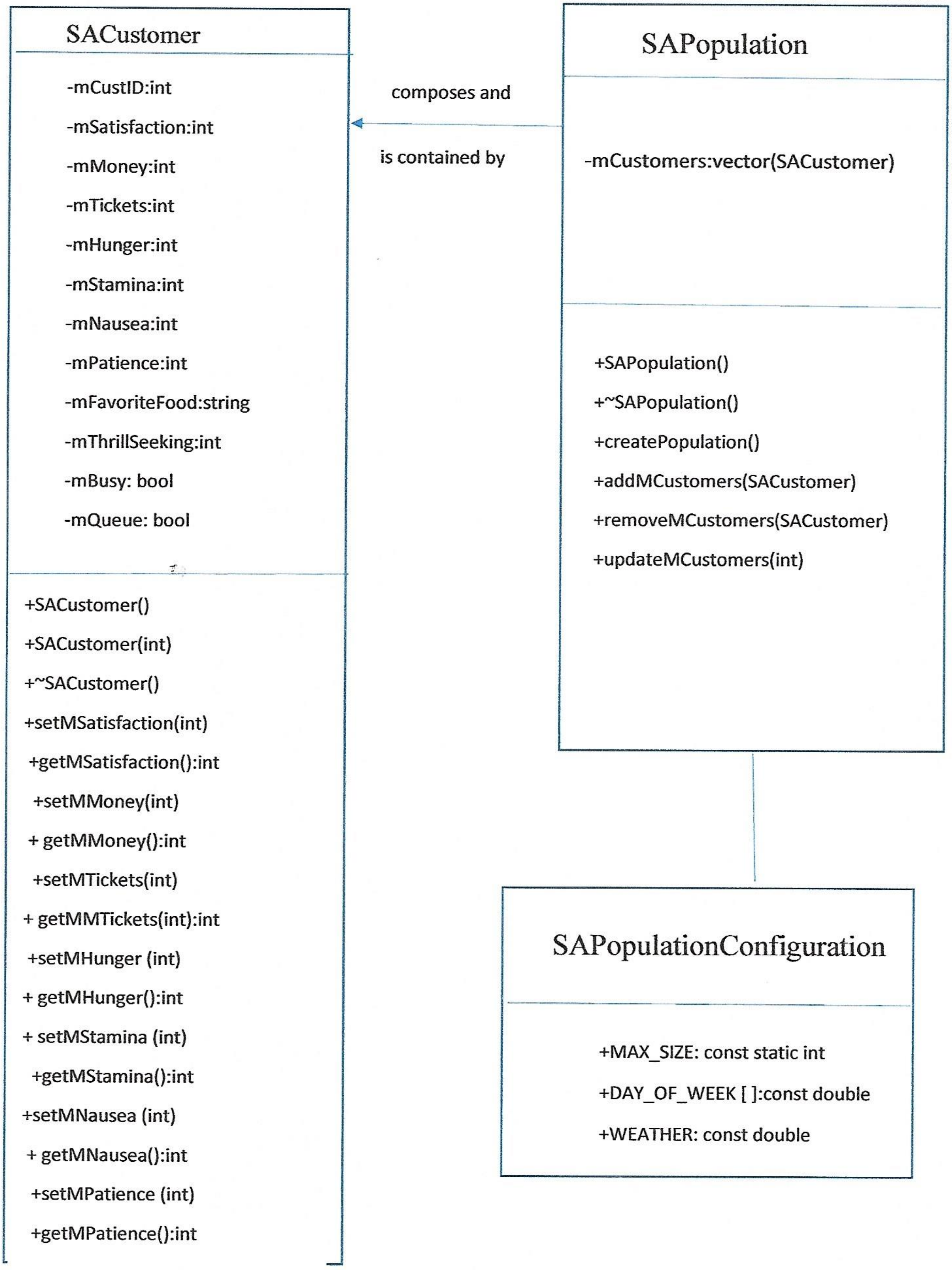
Class diagram

Shrooq Algarni



SACustomer

-mCustID:int
-mSatisfaction:int
-mMoney:int
-mTickets:int
-mHunger:int
-mStamina:int
-mNausea:int
-mPatience:int
-mFavoriteFood:string
-mThrillSeeking:int
-mBusy: bool
-mQueue: bool

+SACustomer()
+SACustomer(int)
+~SACustomer()
+setMSatisfaction(int)
+getMSatisfaction():int
+setMMoney(int)
+ getMMoney():int
+setMTickets(int)
+ getMMTickets(int):int
+setMHunger (int)
+ getMHunger():int
+ setMStamina (int)
+getMStamina():int
+setMNausea (int)
+ getMNausea():int
+setMPatience (int)
+getMPatience():int

SAPopulation

-mCustomers:vector(SACustomer)

+SAPopulation()
+~SAPopulation()
+createPopulation()
+addMCustomers(SACustomer)
+removeMCustomers(SACustomer)
+updateMCustomers(int)

SAPopulationConfiguration

+MAX_SIZE: const static int
+DAY_OF_WEEK [ ]:const double
+WEATHER: const double

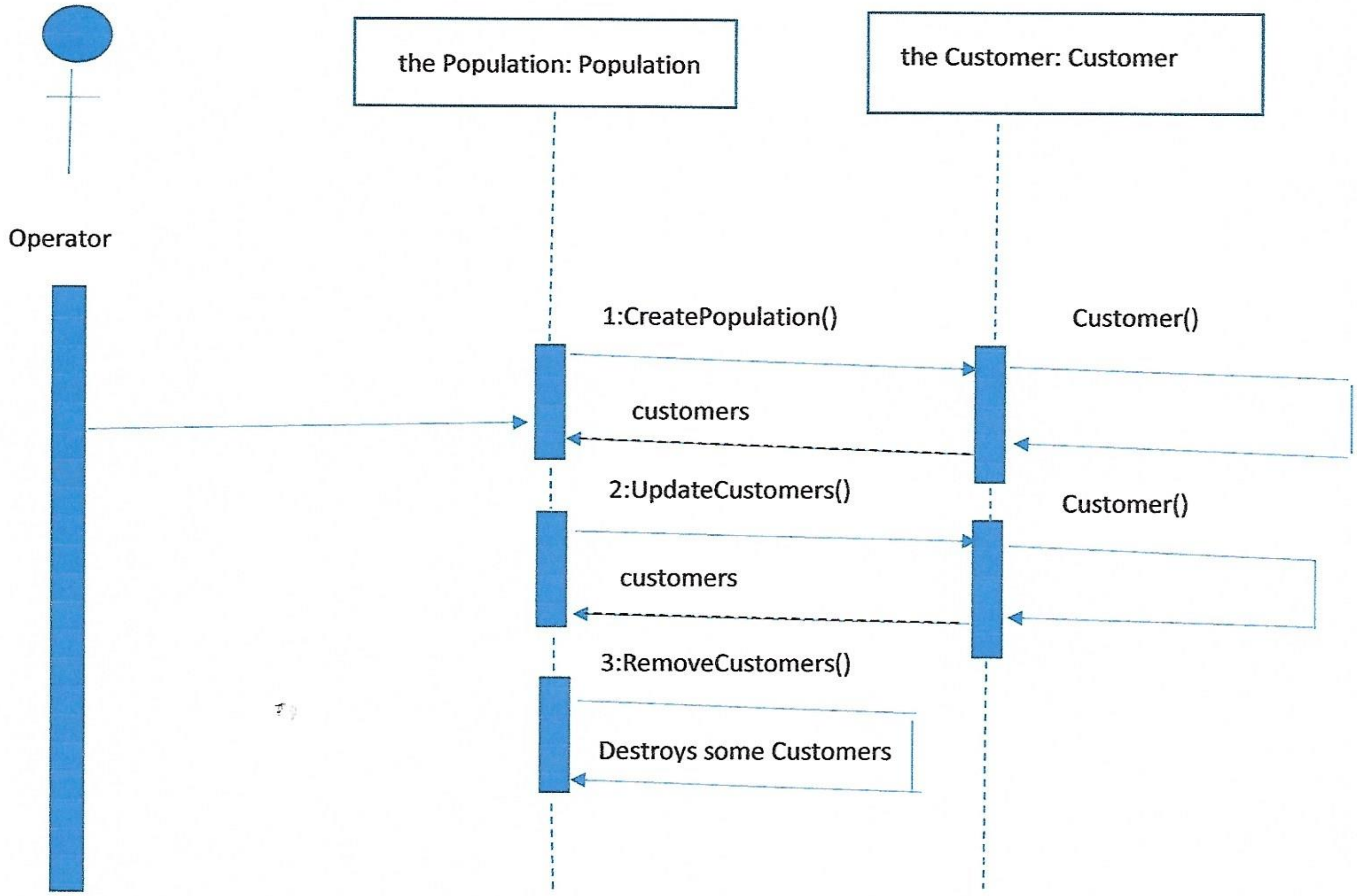
+setMBusy(bool)

+getMBusy():bool

+setMQueue(bool)

+ getMQueue():bool

Sequence diagram



Test cases:

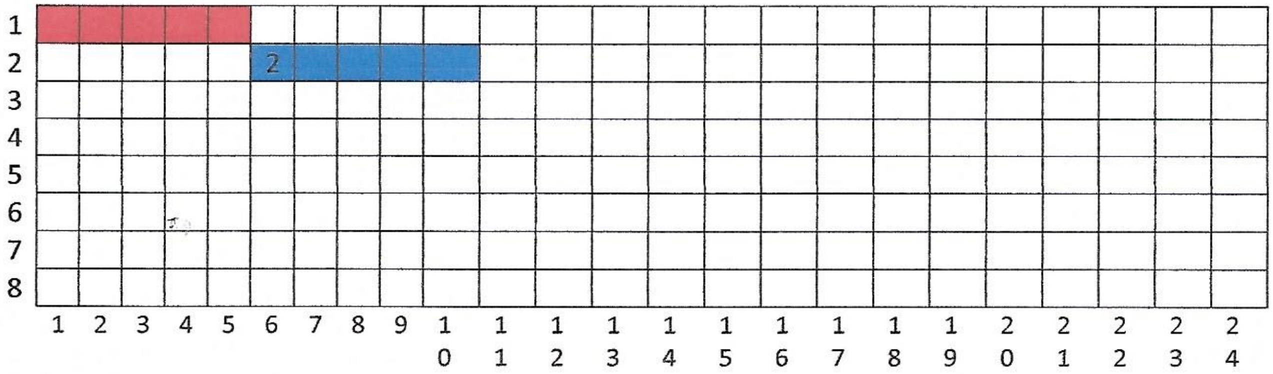
The user should enter the positive number in population size.

The user should not enter number more than the maximum of population size.

The money should not be less than $50.

Gantt time-line:

Task	Duration (Pdays)	Predecessor Task(s)
1. customer	5	-
2. population	5	1



```

#ifndef SA_CUSTOMER_H_
#define SA_CUSTOMER_H_

#include <iostream>
#include<cstring>
using namespace std;
class SACustomer
{
     private :
          int mCustID;
          int mSatisfaction;
          double mMoney;
          int mTickets;
          int mHunger;
          int mStamina;
          int mNausea;
          int mPatience;
          string mFavoriteFood;
          int mThrillSeeking;
          bool mBusy;
          bool mQueue;
     public :
          SACustomer();
          SACustomer(int);
          ~SACustomer();
          void setMSatisfaction(int);
          int getMSatisfaction();
          void setMMoney(int);
          int getMMoney();
          void setMTickets(int);
          int getMTickets();
          void setMHunger (int);
          int getMHunger();
          void setMStamina (int);
          int getMStamina();
          void setMNausea (int);
          int getMNausea();
          void setMPatience (int);
          int getMPatience();
          void setMBusy(bool);
          bool getMBusy();
          void setMQueue(bool);
          bool getMQueue();
};
#endif  // SA_CUSTOMER_H_
```

```
#ifndef SA_POPULATION_H_
#define SA_POPULATION_H_

#include <vector>
#include "customer.h"

class SAPopulation {

private:
    vector<SACustomer> mCustomers;

public:
    SAPopulation();
    ~SAPopulation();
    int createPopulation();
    void addMCustomers(SACustomer);
    void removeMCustomers(SACustomer);
    void updateMCustomers(int);
};
#endif  // SA_POPULATION_H_
```

```
class SAPopulationConfiguration{
   const static int MAX_SIZE = 2000;
   const DOUBLE DAY_OF_WEEK[] = {.7,.2,.2,.2,.2,.2,0.7};  // 0=sun, 1=mon, 2=tue, 3=wed, 4=thu
   const DOUBLE WEATHER[] = {.1,.3,.2,.7,.5};  // 0=snow, 1=cold, 2=rainy, 3=moderate, 4=sunny
};
```

```cpp
#define CATCH_CONFIG_MAIN  // This tells Catch to provide a main() - only do this in one cpp
#include "catch.hpp"
#include <stdlib.h>

#include "customer.h"
#include "population.h"
#include "PopulationConfiguration.h"

TEST_CASE( "Population not negetive", "[population]" ) {
    SAPopulation p;
    REQUIRE( p.createPopulation() >= 0 );
}
TEST_CASE( "Population not more than the maximum of population size", "[population]" ) {
    SAPopulation p;
    REQUIRE( p.createPopulation() <= MAX_SIZE );
}
```