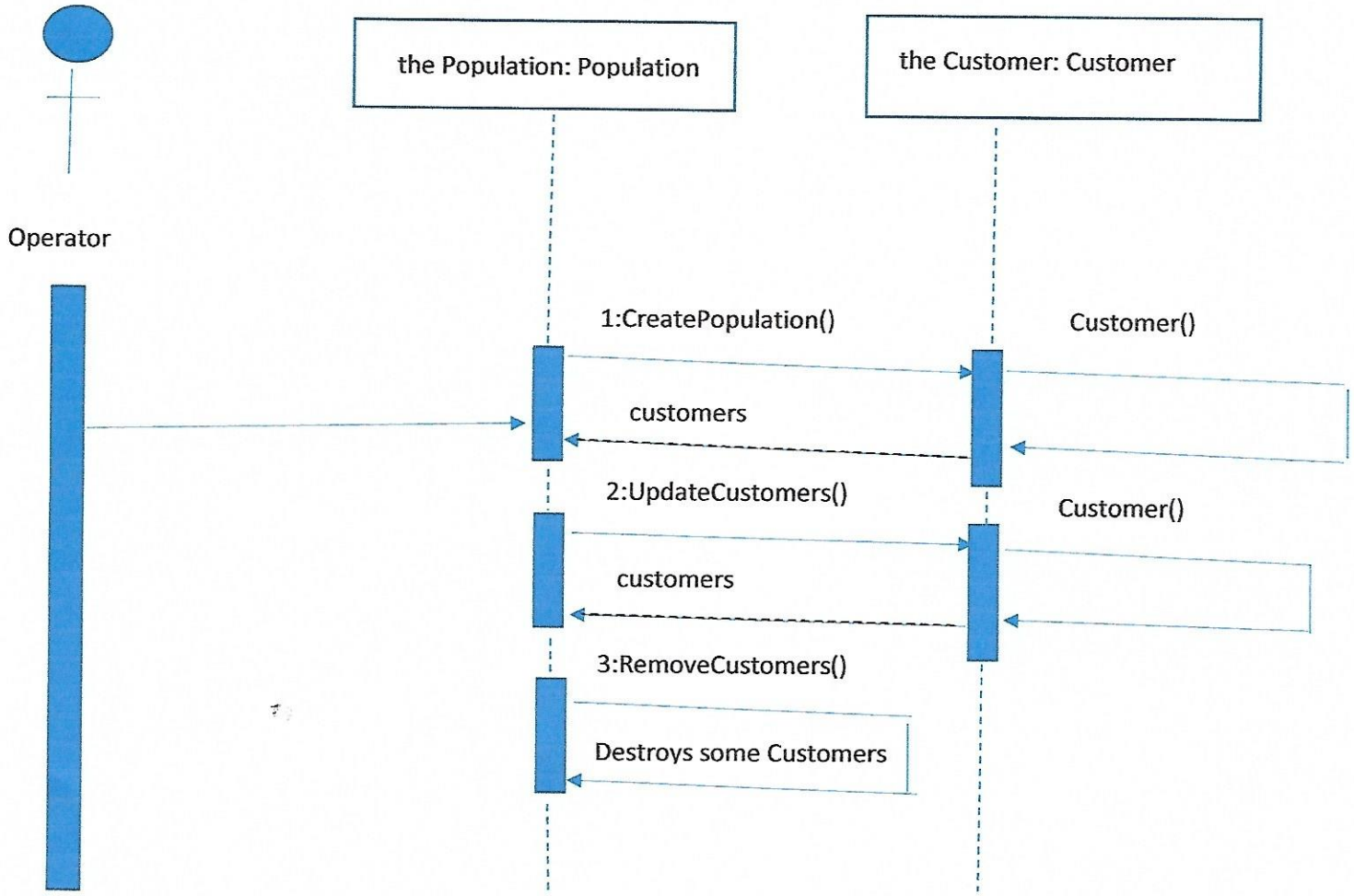
Class diagram

Shrooq Algarni


+setMBusy(bool)
+getMBusy():bool
+setMQueue(bool)
+ getMQueue():bool

Sequence diagram



Test cases:

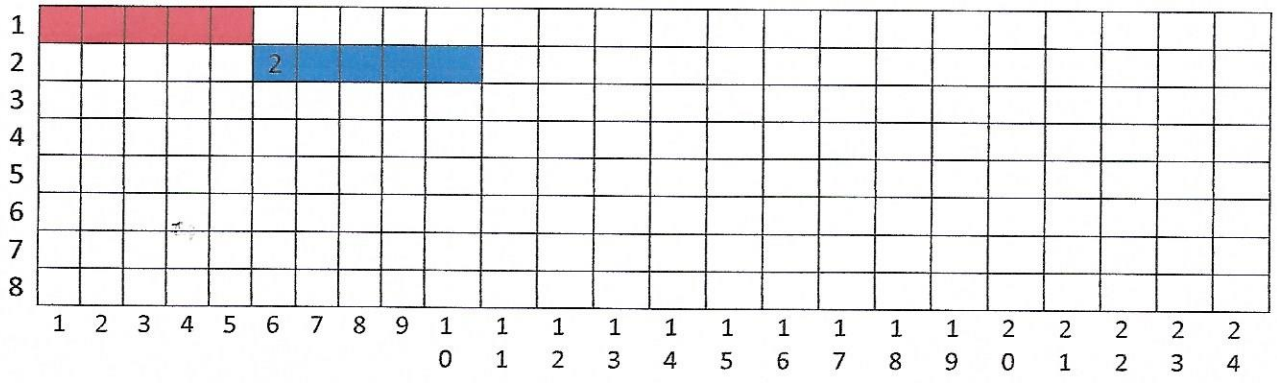
The user should enter the positive number in population size.

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



```
 1:
 2: #ifndef SA_CUSTOMER_H_
 3: #define SA_CUSTOMER_H_
 4:
 5: #include <iostream>
 6: #include<cstring>
 7: using namespace std;
 8: class SACustomer
 9: {
10:      private :
11:           int mCustID;
12:           int mSatisfaction;
13:           double mMoney;
14:           int mTickets;
15:           int mHunger;
16:           int mStamina;
17:           int mNausea;
18:           int mPatience;
19:           string mFavoriteFood;
20:           int mThrillSeeking;
21:           bool mBusy;
22:           bool mQueue;
23:      public :
24:           SACustomer();
25:           SACustomer(int);
26:           ~SACustomer();
27:           void setMCustID(int);
28:           int getMCustID();
29:           void setMSatisfaction(int);
30:           int getMSatisfaction();
31:           void setMMoney(double);
32:           double getMMoney();
33:           void setMTickets(int);
34:           int getMTickets();
35:           void setMHunger (int);
36:           int getMHunger();
37:           void setMStamina (int);
38:           int getMStamina();
39:           void setMNausea (int);
40:           int getMNausea();
41:           void setMPatience (int);
42:           int getMPatience();
43:           void setMBusy(bool);
44:           bool getMBusy();
45:           void setMQueue(bool);
46:           bool getMQueue();
47: };
48: #endif  // SA_CUSTOMER_H_
```

```
 1: #ifndef SA_POPULATION_H_
 2: #define SA_POPULATION_H_
 3:
 4: #include <vector>
 5: #include "customer.h"
 6:
 7: class SAPopulation {
 8:
 9: private:
10:     int mPopSize;
11:     vector<SACustomer> mCustomers;
12:
13: public:
14:     SAPopulation();
15:     ~SAPopulation();
16:     void setMPopSize(int);
17:     int getMPopSize();
18:     void setMCustomers(vector<SACustomer>);
19:     vector<SACustomer> getMCustomers();
20:     int createPopulation();
21:     void addMCustomers(SACustomer);
22:     void removeMCustomers(SACustomer);
23:     void updateMCustomers(int);
24: };
25: #endif  // SA_POPULATION_H_
```

```
1: #ifndef SA_CONSTANT_H_
2: #define SA_CONSTANT_H_
3: const static int MAX_SIZE = 2000;
4: const double DAY_OF_WEEK[] = {0.7,0.2,0.2,0.2,0.2,0.2,0.7};  // 0=sun, 1=mon, 2=tue, 3=wed, 4=
5: const double WEATHER[] = {0.1,0.3,0.2,0.7,0.5};  // 0=snow, 1=cold, 2=rainy, 3=moderate, 4=sun
6: #endif  // SA_CONSTANT_H_
```

```cpp
#include <iostream>

#include "../../inc/SA/customer.h"

using namespace std;

SACustomer::SACustomer(){
  cout << "SACustomer() Constructor" << endl;
}

SACustomer::SACustomer(int id){
  cout << "SACustomer(int) Constructor" << endl;
  mCustID = id;
}

SACustomer::~SACustomer(){
  cout << "~SACustomer() Destructor" << endl;
}
void SACustomer::setMCustID(int c) {
    cout << "SACustomer::setMCustID(int)" << endl;
    mCustID = c;
}

int SACustomer::getMCustID() {
    cout << "SACustomer::getMCustID()" << endl;
    return  mCustID;
}

void SACustomer::setMSatisfaction(int s) {
    cout << "SACustomer::setMSatisfaction(int)" << endl;
    mSatisfaction = s;
}

int SACustomer::getMSatisfaction() {
    cout << "SACustomer::getMSatisfaction()" << endl;
    return mSatisfaction;
}
void SACustomer::setMMoney(double m) {
    cout << "SACustomer::setMMoney(double)" << endl;
    mMoney= m;
}

double SACustomer::getMMoney() {
    cout << "SACustomer::getMMoney()" << endl;
    return mMoney;
}
void SACustomer::setMTickets(int t) {
    cout << "SACustomer::setMTickets(int)" << endl;
    mTickets = t;
}

int SACustomer::getMTickets() {
    cout << "SACustomer::getMTickets()" << endl;
    return mTickets;
}
```

```
56: void SACustomer::setMHunger(int h) {
57:     cout << "SACustomer::setMHunger(int)" << endl;
58:     mHunger = h;
59: }
60:
61: int SACustomer::getMHunger() {
62:     cout << "SACustomer::getMHunger()" << endl;
63:     return mHunger;
64: }
65: void SACustomer::setMStamina(int a) {
66:     cout << "SACustomer::setMStamina(int)" << endl;
67:     mStamina = a;
68: }
69:
70: int SACustomer::getMStamina() {
71:     cout << "SACustomer::getMStamina()" << endl;
72:     return  mStamina;
73: }
74: void SACustomer::setMNausea(int n) {
75:     cout << "SACustomer::setMNausea(int)" << endl;
76:     mNausea = n;
77: }
78:
79: int SACustomer::getMNausea() {
80:     cout << "SACustomer::getMNausea()" << endl;
81:     return  mNausea;
82: }
83: void SACustomer::setMPatience(int p) {
84:     cout << "SACustomer::setMPatience(int)" << endl;
85:     mPatience = p;
86: }
87:
88: int SACustomer::getMPatience() {
89:     cout << "SACustomer::getMPatience()" << endl;
90:     return  mPatience;
91: }
92: void SACustomer::setMBusy(bool b) {
93:     cout << "SACustomer::setMBusy(bool)" << endl;
94:     mBusy = b;
95: }
96:
97: bool SACustomer::getMBusy() {
98:     cout << "SACustomer::getMBusy()" << endl;
99:     return  mBusy;
100: }
101: void SACustomer::setMQueue(bool q) {
102:     cout << "SACustomer::setMQueue(bool)" << endl;
103:     mQueue = q;
104: }
105:
106: bool SACustomer::getMQueue() {
107:     cout << "SACustomer::getMQueue()" << endl;
108:     return   mQueue;
109: }
```

```cpp
#include <iostream>

#include "../../inc/SA/population.h"
#include "PopulationConfiguration.cpp"
using namespace std;



SAPopulation::SAPopulation(){
  cout << "SAPopulation() Constructor."<< endl;
}

SAPopulation::  ~SAPopulation(){
  cout << "~SAPopulation() Destructor."<< endl;
}
void SAPopulation::setMPopSize(int i) {
    cout << "SACustomer::setMPopSize(int)" << endl;
    mPopSize = i;
}

int SAPopulation::getMPopSize() {
    cout << "SACustomer::getMPopSize()" << endl;
    return  mPopSize;
}
void SAPopulation::setMCustomers(vector<SACustomer> x){
    cout << "SAPopulation::setMCustomers(vector<SACustomer>)" << endl;
    mCustomers=x;
}
vector<SACustomer> SAPopulation::getMCustomers(){
    cout << "SAPopulation::getMCustomers()" << endl;
    return mCustomers;
}
int SAPopulation::createPopulation() {
  cout << "SAPopulation::createPopulation()"<< endl ;
  int initialSize = 0;
  initialSize = SAPopulationConfig::generateSize();
  return initialSize;
}

void SAPopulation::addMCustomers(SACustomer cust){
  cout << "SAPopulation::addMCustomers(SACustomer)"<< endl ;
  mCustomers.push_back (cust);
}


 void SAPopulation::removeMCustomers(SACustomer cust){
  cout << "SAPopulation::removeMCustomers(SACustomer)"<< endl;
  //mCustomers.erase (mCustomers.begin() + 5);  // think logic to remove from vector
  /*int index = 0;
  for (std::vector<SACustomer>::iterator it = mCustomers.begin() ; it != mCustomers.en
    SACustomer c = *it;
    if(cust.getMCustID() == c.getMCustID()) {
        index =
    }*/
}
```

```
56:
57: void SAPopulation::updateMCustomers(int cust){
58:
59:   cout << "SAPopulation::updateMCustomers(int)"<< endl;
60:
61: }
62:
63:
64:
```

```cpp
#include <iostream>
#include <cstdlib>    /* srand, rand */
#include <ctime>      /* time */

#include "../../inc/SA/constant.h"

using namespace std;

class SAPopulationConfig {
public:
    static int generateSize() {
        int weatherIndex = (int) (rand() / 1000) % 5;
        cout << "weatherIndex: " << weatherIndex << endl;
        int dayOfWeekIndex = (int) (rand() / 1000) % 7;
        cout << "dayOfWeekIndex: " << dayOfWeekIndex << endl;

        return MAX_SIZE * DAY_OF_WEEK[dayOfWeekIndex] * WEATHER[weatherIndex];
    }
};

/**
* Just for test purpose
*/
/*int main() {
    srand (time(0));

    int size = SAPopulationConfig::generateSize();
    cout << "SIZE: " << size << endl;
    return 0;
}*/
```

```
 1: #define CATCH_CONFIG_MAIN  // This tells Catch to provide a main() - only do this in one cpp
 2: #include "catch.hpp"
 3: #include <stdlib.h>
 4:
 5: //#include "../../inc/SA/customer.h"
 6: //#include "../../inc/SA/population.h"
 7: //#include "../../src/SA/PopulationConfiguration.cpp"
 8: #include "../../src/SA/customer.cpp"
 9: #include "../../src/SA/population.cpp"
10:
11: TEST_CASE( "Population not negetive", "[population]" ) {
12:     SAPopulation p;
13:     REQUIRE( p.createPopulation() >= 0 );
14: }
15:
16: TEST_CASE( "Population not more than the maximum of population size", "[population]" ) {
17:     SAPopulation p;
18:     REQUIRE( p.createPopulation() <= MAX_SIZE );
19: }
```