Name:haris Farooq

Class bscs

Section 5d

Roll number 21f-9064

#include <iostream>

#include <thread>

#include <mutex>

#include <condition\_variable>

#include <cstdlib>

#include <random>

#include <chrono>

using namespace std;

const int row = 5;

const int col = 5;

int Ingcount = 0;

class Dispenser {

mutex mtx;

char disp[row][col];

char prev;

char selectedIngredients[row \* col];

public:

Dispenser() {

char ing[4] = { 'R', 'Y', 'b', 'B' };

std::random\_device rd;

std::mt19937 gen(rd());

std::uniform\_int\_distribution<> dis(0, 3);

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

disp[i][j] = ing[dis(gen)];

}

}

}

char\* SelectIng(int x, int y) {

std::lock\_guard<std::mutex> lock(mtx);

selectedIngredients[Ingcount] = disp[x][y];

prev = disp[x][y];

disp[x][y] = '-';

RollBack();

Ingcount++;

checkExplosion(x, y);

return selectedIngredients;

}

void DispenserDisplay() {

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

cout << disp[i][j] << "\t";

}

cout << endl;

}

}

void RollBack() {

for (int i = 0; i < row - 1; i++) {

for (int j = 0; j < col; j++) {

if (disp[i][j] == '-') {

char temp = disp[i][j];

disp[i][j] = disp[i + 1][j];

disp[i + 1][j] = temp;

}

}

}

}

void checkExplosion(int x, int y) {

if (disp[x][y] == '-') {

return;

} else if (x > 0 && x < 4) {

if (disp[x][y] == disp[x + 1][y]) {

SelectIng(x + 1, y);

}

if (disp[x][y] == disp[x - 1][y]) {

SelectIng(x - 1, y);

}

if (disp[x][y] == prev) {

SelectIng(x, y);

}

} else if (x == 0 && disp[x][y] == prev) {

SelectIng(x, y);

} else {

return;

}

}

void returnToDispenser(char\* ingredientSelected) {

std::random\_device rd;

std::mt19937 gen(rd());

std::uniform\_int\_distribution<> dis(0, 3);

while (Ingcount > 0) {

int tempx = dis(gen);

int tempy = dis(gen);

if (disp[tempx][tempy] == '-') {

disp[tempx][tempy] = ingredientSelected[Ingcount - 1];

Ingcount--;

RollBack();

}

}

}

};

struct Styles {

int black = 0;

int red = 0;

int blue = 0;

int yellow = 0;

};

class potionTiles {

mutex mtx;

Styles\* Sty;

Styles CompletedPotion[5];

int potionRec[5] = { 0, 0, 0, 0, 0 };

Styles\* selected;

int CompletedCount = 0;

public:

int potionCount = -1;

potionTiles() {

Sty = new Styles[5];

Sty[0].black = 3, Sty[0].blue = 1, Sty[0].red = 0, Sty[0].yellow = 2;

Sty[1].black = 0, Sty[1].blue = 1, Sty[1].red = 2, Sty[1].yellow = 1;

Sty[2].black = 2, Sty[2].blue = 0, Sty[2].red = 1, Sty[2].yellow = 2;

Sty[3].black = 2, Sty[3].blue = 2, Sty[3].red = 0, Sty[3].yellow = 0;

Sty[4].black = 0, Sty[4].blue = 1, Sty[4].red = 2, Sty[4].yellow = 2;

selected = new Styles[2];

}

void DisplayAllPotion() {

cout << "All 5 styles are : \n";

for (int i = 0; i < 5; i++) {

cout << "Style " << i + 1 << " : " << "\tblack : " << Sty[i].black << "\tblue : " << Sty[i].blue << "\tRed : " << Sty[i].red << "\tYellow : " << Sty[i].yellow << endl;

}

}

void placePotionTiles() {

std::lock\_guard<std::mutex> lock(mtx);

if (potionCount >= 1) {

cout << "No more potion tiles can be placed";

return;

} else {

DisplayAllPotion();

int choice = 0;

while (1) {

cout << "Enter your choice : ";

cin >> choice;

if (choice > 0 && choice < 6) {

cout << "That's a valid choice.";

potionCount++;

potionRec[choice - 1] = 1;

return;

} else {

cout << "Invalid choice";

}

}

}

}

bool CompletedPotions() {

bool flag = false;

for (int i = 0, j = 0; i < row && j < 2; i++) {

if (selected[j].black == Sty[i].black && selected[j].blue == Sty[i].blue && selected[j].red == Sty[i].red && selected[j].yellow == Sty[i].yellow) {

CompletedPotion[CompletedCount] = selected[j];

Styles comptemp = selected[j];

selected[j] = selected[j - 1];

selected[j - 1] = comptemp;

potionRec[i] = 0;

potionCount--;

j--;

i = 0;

flag = true;

}

}

return flag;

}

bool placeIng(char ingredientSelected) {

int ingredients[4];

cout << "Place ingredients on previously selected potion tiles or place them in flask.";

cout << "\n Presiously selected potions are : \n";

int j = 1;

for (int i = 0; i < 5; i++) {

if (potionRec[i] == 1) {

cout << "Choice : " << j << endl;

cout << "Style " << i + 1 << " : " << "\tblack : " << Sty[i].black << "\tblue : " << Sty[i].blue << "\tRed : " << Sty[i].red << "\tYellow : " << Sty[i].yellow << endl;

j++;

}

}

if (j == 1) {

cout << "Choice 2 == Choice 1" << endl;

}

int temp;

while (1) {

cout << "Enter your choice ( 1 or 2 ) : ";

cin >> temp;

if (temp == 1 or temp == 2)

break;

else

cout << "Invalid choice";

}

bool flag = false;

int check = -1;

for (int i = 0; i < row; i++) {

if (potionRec[i] == 1) {

check++;

if (check == temp - 1) {

if (selected[temp - 1].black < Sty[i].black) {

if (ingredientSelected == 'B') {

selected[temp - 1].black++;

cout << "Ingredient is placed on potion tile" << endl;

flag = true;

}

}

if (selected[temp - 1].blue < Sty[i].blue) {

if (ingredientSelected == 'b') {

selected[temp - 1].blue++;

cout << "Ingredient is placed on potion tile" << endl;

flag = true;

}

}

if (selected[temp - 1].red < Sty[i].red) {

if (ingredientSelected == 'R') {

selected[temp - 1].red++;

cout << "Ingredient is placed on potion tile" << endl;

flag = true;

}

}

if (selected[temp - 1].yellow < Sty[i].yellow) {

if (ingredientSelected == 'Y') {

selected[temp - 1].yellow++;

cout << "Ingredient is placed on potion tile" << endl;

flag = true;

}

}

}

}

}

for (int i = 0, j = 0; i < 5; i++) {

if (potionRec[i] == 1) {

cout << "Ingredients need to complete Choice " << j + 1 << " : ";

cout << "Style " << i + 1 << " : " << "\tblack : " << Sty[i].black - selected[j].black << "\tblue : " << Sty[i].blue - selected[j].blue << "\tRed : " << Sty[i].red - selected[j].red << "\tYellow : " << Sty[i].yellow - selected[j].yellow << endl;

j++;

}

}

if (flag == false) {

cout << "Sorry Ingredient can't be placed on selected potion tile" << endl;

}

return flag;

}

};

class flask {

mutex mtx;

char ing[3];

int count = 0;

public:

flask() {}

bool placeInFlask(char ingredient) {

std::lock\_guard<std::mutex> lock(mtx);

if (count < 3) {

ing[count] = ingredient;

count++;

cout << "Ingredient placed in Flask.";

return true;

} else {

cout << "Sorry! No more ingredients can be placed in Flask.";

return false;

}

}

void DisplayFlask() {

for (int i = 0; i < 3; i++) {

cout << i << ". " << ing[i] << endl;

}

}

};

class Gameboard {

mutex gameMutex;

condition\_variable turnCV;

bool isPlayer1Turn = true;

Dispenser D;

potionTiles Tiles;

flask Extra;

public:

Gameboard() {}

void playerAction(int playerId) {

unique\_lock<mutex> lock(gameMutex);

while (true) {

turnCV.wait(lock, [this, playerId]() { return (playerId == 1) == isPlayer1Turn; });

int temp;

if (Tiles.potionCount == 1) {

if (Tiles.CompletedPotions()) {

Tiles.placePotionTiles();

} else {

int x, y;

D.DispenserDisplay();

cout << "Enter (x,y):";

cin >> x >> y;

char\* ingredientSelected = D.SelectIng(x, y);

D.DispenserDisplay();

int select;

int choice;

cout << "\n1.Place.\n2.Return to Dispenser.\nYou want to place ingredients or return remaining ingredients : ";

cin >> choice;

if (choice == 1) {

while (Ingcount != 0) {

for (int i = 0; i < Ingcount; i++) {

cout << i << "." << ingredientSelected[i] << endl;

}

cout << "\nEnter selected ingredient (example: 1): ";

cin >> select;

int temp;

cout << "\n1.Place on potion Tile.\n2.Place in flask.\nYou want to place ingredients or return remaining ingredients : ";

cin >> temp;

if (temp == 1) {

if (Tiles.placeIng(ingredientSelected[select])) {

ingredientSelected[select] = '-';

for (int i = 0; i < row - 1; i++) {

if (ingredientSelected[i] == '-') {

char temp = ingredientSelected[i];

ingredientSelected[i] = ingredientSelected[i + 1];

ingredientSelected[i + 1] = temp;

}

}

Ingcount--;

}

} else if (temp == 2) {

if (Extra.placeInFlask(ingredientSelected[select])) {

ingredientSelected[select] = '-';

for (int i = 0; i < row - 1; i++) {

if (ingredientSelected[i] == '-') {

char temp = ingredientSelected[i];

ingredientSelected[i] = ingredientSelected[i + 1];

ingredientSelected[i + 1] = temp;

}

}

Ingcount--;

}

}

}

} else if (choice == 2) {

D.returnToDispenser(ingredientSelected);

}

}

} else {

Tiles.placePotionTiles();

}

isPlayer1Turn = !isPlayer1Turn;

turnCV.notify\_all();

}

}

void startGame() {

thread player1(&Gameboard::playerAction, this, 1);

thread player2(&Gameboard::playerAction, this, 2);

player1.join();

player2.join();

}

};

int main() {

unsigned seed = chrono::system\_clock::now().time\_since\_epoch().count();

default\_random\_engine generator(seed);

srand(time(nullptr));

Gameboard game;

game.startGame();

return 0;

}