עבודת גמר C++

**תוכנה לניהול פרויקטים**

**שנה ב' סמסטר קיץ תש"פ**

**מגישים:**

|  |  |
| --- | --- |
| **דוד מוסייב** | **319455119** |
| **יקיר מימון** | **203987557** |
| **שקד ספקטור** | **308132281** |

**מרצים**

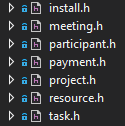
**מר נדב וולך, גב' מעיין זנו**

**תאריך: 24.09.2020**

**מבוא ורקע**

1. ניהול פרויקטים הינו אתגר אשר מהווה אתגר ניהולי ואתגר טכני.
2. על כן התוכנה שאנחנו מציגים בפרויקט זה , מאחסנת ומציגה נתונים על הפרויקט מתחילתו ועד סופו, לפי הסדר שהוזנו על ידי המשתמש.
3. המשתמש יכניס לתוך התוכנה את שם הפרויקט, שמות המפגשים וסוגי המפגשים בפרויקט , את כמות הכסף הנדרש אם נדרש , את פרטי המשתתפים ואת המשאבים הנדרשים עבור התקנות של תכולות הפרויקט.
4. **מבנה התוכנה והמחלקות**

רשימת קבצי הHEADER של המחלקות:



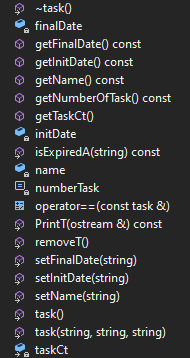
להלן פירוט והסבר של כל מחלקה:

1. מחלקת task – מחלקה האב של מחלקות install וmeeting "סבא" של מחלקת payment . מחזיקה שדות :

|  |  |  |
| --- | --- | --- |
| **תאריך התחלה string initDate** | **תאריך סוף string  finalDate** | **שם string name** |

מטרת המחלקה להוות אוביקט בסיס למשימה שתוכל בתוך פרויקט.

להלן רשימה סה"כ פונקציות שהמחלקה מכילה:



1. מחלקת meeting -מחלקת ילד של task ומחלקת אב של payment.

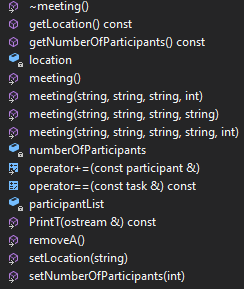
המחלקה נעזרת במחלקת עזר participant.

בנוסף לשדות של task מחזיקה את השדות:

|  |  |
| --- | --- |
| **משתתפים  participant \* participant** | **מיקום string location** |
| **כמות משתתפים int numOfParticipants** |  |

מטרת המחלקה להוות אוביקט פגישה שמצביע על נתוני משתתפי הפגישה ממחלקת particpant ומחזיק נתון על מיקום הפגישה.

להלן רשימה סה"כ פונקציות שהמחלקה מכילה:



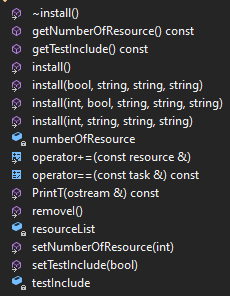
1. מחלקת install – מחלקת ילד של task , נעזרת במחלקת עזר של resource .

בנוסף לשדות של מחלקת task :

|  |  |
| --- | --- |
| **כמות סוגי משאבים intNumberOfResource** | **הפנייה למשאב resource\* listOfResource** |
| **כולל בדיקה bool test** |  |

מטרת המחלקה היא ליצור אוביקט מסוג התקנה, בנוסף מקשר לכמות וסוג המשאבים הנדרשים עבור ההתקנה

להלן רשימה סה"כ הפונקציות שהמחלקה מכילה:

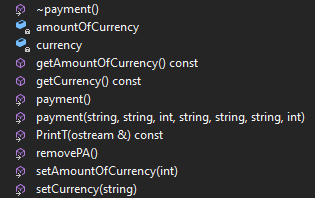


1. מחלקת payment – מחלקת ילד של meeting , בנוסף לשדות של מחלקת task וmeeting:

**כמות הכסף שנדרש לתשלום  
int amountPay**

מטרת המחלקה היא ליצור אוביקט של פגישה שבנוסף לפגישה נדרש בה גם תשלום כספי.

להלן רשימה של סה"כ הפונקציות שהמחלקה מכילה:

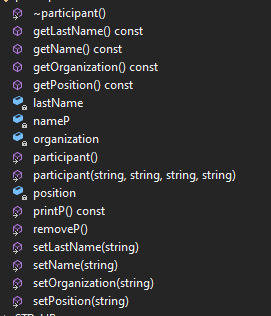


1. מחלקת participant , מחלקה שמחזיקה נתונים אודות המשתתפים בפגישות, מחזיקה בשדות:

|  |  |  |  |
| --- | --- | --- | --- |
| **שם המשתתף string name** | **שם משפחה המשתתף string lastName** | **ארגון string organization** | **תפקיד string position** |

מטרת המחלקה היא להחזיק נתונים אודות כלל המשתתפים בפגישות של הפרויקט.

להלן רשימה של סה"כ הפונקציות שהמחלקה מכילה:

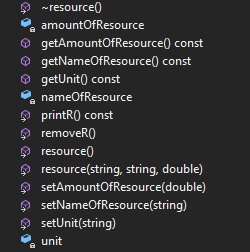


1. מחלקת resource, מחלקה שמחזיקה נתונים אודות המשאבים הנדרשים לכל התקנה . מחזיקה בשדות:

|  |  |  |
| --- | --- | --- |
| **כמות משאב amountOfReasource** | **שם המשאב nameOfResource** | **שם גודל המשאב string nameOfResource** |

מטרת המחלקה להחזיק נתונים אודות כלל המשאבים הדרושים עבור התקנת התוכלות בפרויקט.

להלן רשימה של סה"כ הפונקציות שהמחלקה מכילה:



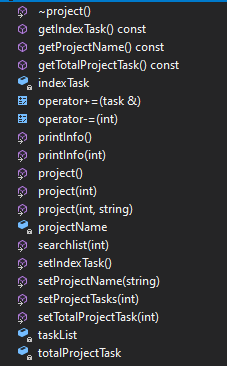
1. מחלקת project , מחלקה שמכילה בתוכה כמות אובייקטים מסוג task , אשר יכולים להיות או install או meeting או payment.

המחלקה מחזיקה בשדות:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **מספר משימה const in numberTask** | **כמות משימות בפרויקט int totalTask** | **שם הפרויקט string projectName** | **אינדקס נוכחי int indexTask** | **רשימת מטלות task \*\*tasklist** |
| **מספר הפרויקט**  **const int numberProject;** | **סופר פרויקטים**  **static int projCt;** |  |  |  |

מטרת המחלקה היא לאגד לתוכה מכלול של משימות לפי הזנה של המשתתף ובעצם לאפשר למשתמש גישה לפונקציות אשר נותנות מבט כולל על הפרויקט.

להלן רשימה של סה"כ הפונקציות שהמחלקה מכילה:



**טבלה מסכמת של כלל הפונקציות**

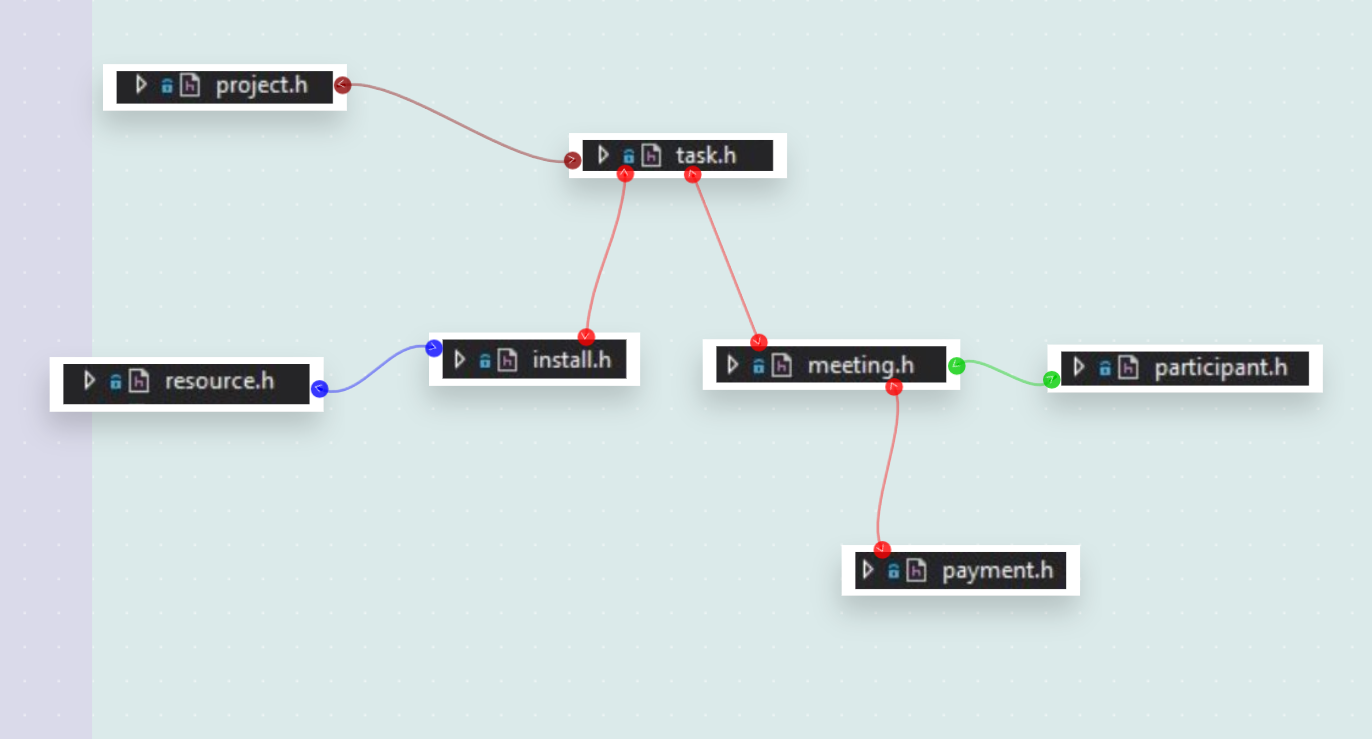
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **מחלקה** | **תיאור** | **תאריך התחלה string initDate** | **תאריך סוף string  finalDate** | **שם string name** | **משתתפים  string\* participant** | **מיקום string location** | **כמות הכסף שנדרש לתשלום int amountPay** | **כולל בדיקה bool test** | **הפנייה לפגישה access\* accessReffer** | **הפנייה להתקנה install\* installReffer** | **מספר משימה const in numberTask** | **סופר משימות static int taskCout** | **כמות משימות בפרויקט int totalTask** | **כמות סוגי משאבים intNumberOfResource** | **הפנייה למשאב resource\* listOfResource** | **כמות משאב amountOfReasource** | **שם המשאב nameOfResource** | **שם גודל המשאב string nameOfResource** | **שם המשתתף string name** | **שם משפחה המשתתף string lastName** | **ארגון string organization** | **תפקיד string position** | **כמות משתתפים int numOfParticipants** | **שם הפרויקט string projectName** | **אינדקס נוכחי int indexTask** | **רשימת מטלות task \*\*tasklist** |
| 1 | משימה task | **V** | **V** | **V** | X | X | X | X | X | X | **V** | **V** | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 2 | פגישה meeting | **\*** | **\*** | **\*** | **V** | **V** | X | X | X | X | **\*** | **\*** | X | X | X | X | X | X | X | X | X | X | **V** | X | X | X |
| 4 | פגישהXתשלום payment | **\*** | **\*** | **\*** | **\*** | **\*** | **V** | X | X | **V** | **\*** | **\*** | X | X | X | X | X | X | X | X | X | X | **\*** | X | X | X |
| 3 | התקנהinstall | **\*** | **\*** | **\*** | X | X | X | **V** | **V** | X | **\*** | **\*** | X | **V** | **V** | X | X | X | X | X | X | X | X | X | X | X |
| 5 | פרויקט project | X | X | X | X | X | X | X | X | X | **V** | X | **V** | X | X | X | X | X | X | X | X | X | X | **V** | **V** | **V** |
| 6 | משאב resource | X | X | X | X | X | X | X | X | X | X | X | X | X | X | **V** | **V** | **V** | X | X | X | X | X | X | X | X |
| 7 | משתתף participant | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | **V** | **V** | **V** | **V** | X | X | X | X |

**V -קיים במחלקה**

**\*- ירושה**

**X- לא קיים במחלקה**

**תלות בין המחלקות**

****

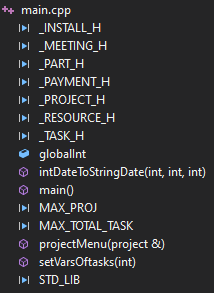
באיור ניתן לראות את הקשר בין המחלקות השונות כפי שתוארו בתחילת המסמך.

קשר מאותו צבע מתאר ירושה, שאר הקשר מתארים שימוש במחלקה השנייה.

קשר בצבעים ירוק וכחול מתארים קשר שימוש של מחלקת install וmeeting בresource וparticipant בהתאמה.

קשר בצבע בורדו מתאר קשר שימוש של מחלקת project במחלקה האבסטרקטית task.

**מבנה הMAIN**



הMain בנוי ממספר חלקים, בחלק הראשון ניתן לבצע הדגמה של בניית 3 פרוייקטים עם 4 משימות כאשר בכל משימה נוספים או משתתפים (participants) או משאבים (resources) ההוספה מתבצעת דרך אופרטור += וביצוע downcast מאובייקט האב task.

בחלק השני קיים תפריט ראשי המאפשר למשתמש לבנות אחד מהבאים,(המספור לפי הבחירה בתפריט):

1. פרויקט ריק עם 0 משימות.
2. פרויקט עם מספר משימות שהמשתמש יבחר.
3. פרויקט עם שם ומספר משימות שהמשתמש יבחר.

לאחר יצירת הפרויקט המשתמש יקבל הודעה שמעניקה לו את שם הפרויקט בברית המחדל שנוצר ואת מספר הפרויקט .

במידה ויבחר המשתמש לנהל את הפרויקט , להוסיף משימות ניתן לבצע זאת על ידי בחירת באפשרות 4 ו5 , אשר מוצאות את הפרויקט המדובר לפי השם הייחודי של הפרויקט (הראשון שימצא) או לפי מספר הייחודי שניתן לכל פרויקט.

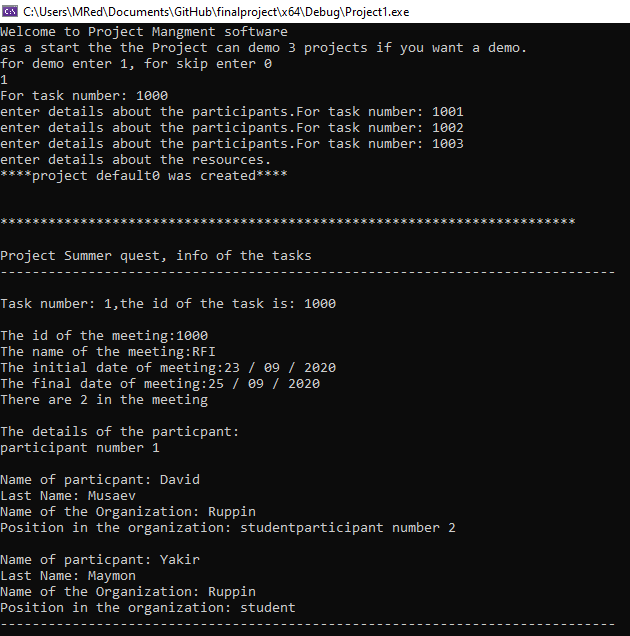
בנוסף ניתן להציג את כלל הפרויקטים אשר נפתחו ואת פרטיהם.

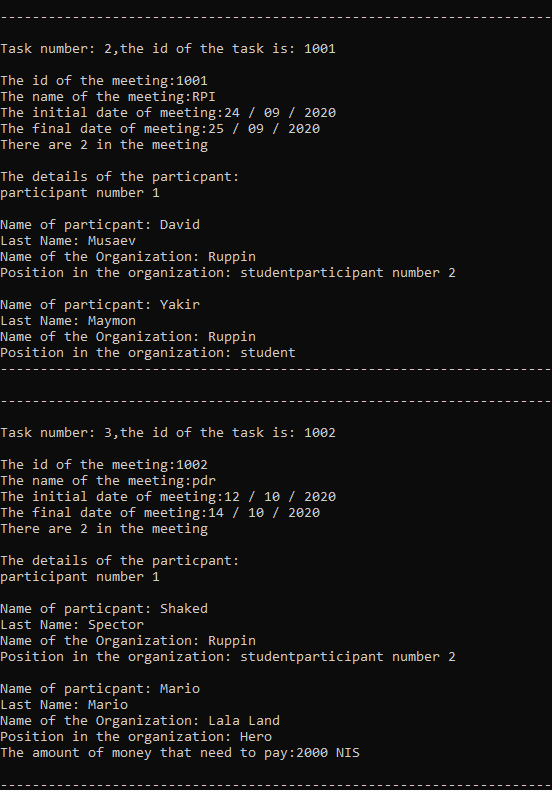
לאחר מכן המשתמש יכנס לתפריט נוסף שבו יהיה ניתן להוסיף, למחוק , להדפיס ,לשנות שם ולמחוק את הפרויקט.

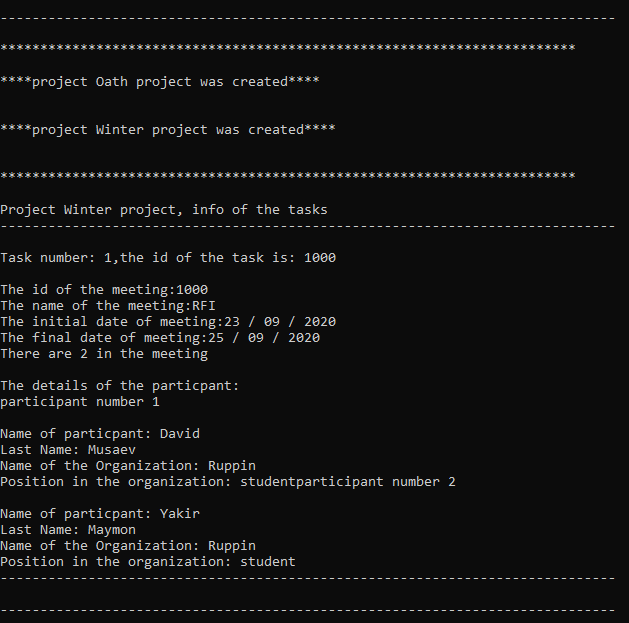
כלל המשימות מנוהלות במשתנה גלובלי בMAIN שניתן להגביל את כמות המשימות אשר בשימוש,

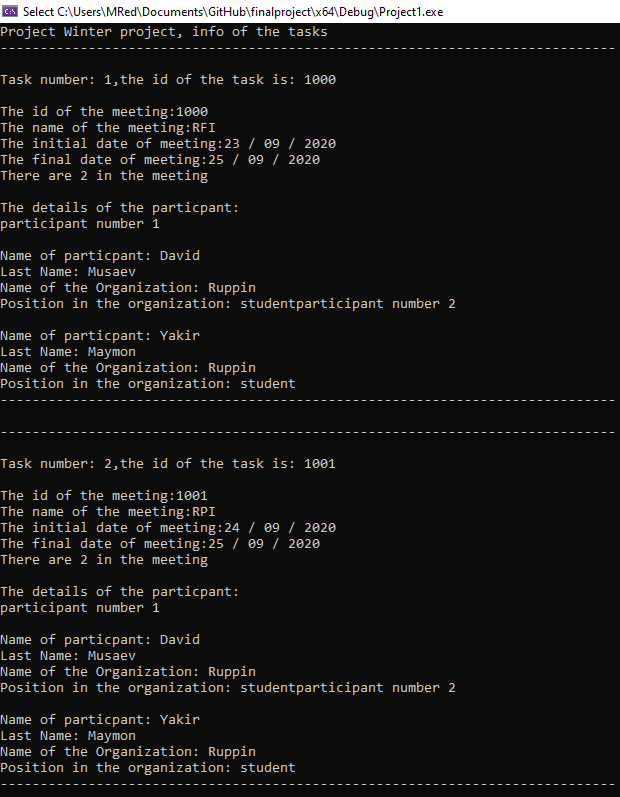
באותו אופן הוגבלו כמות הפרויקטים הפתוחים.

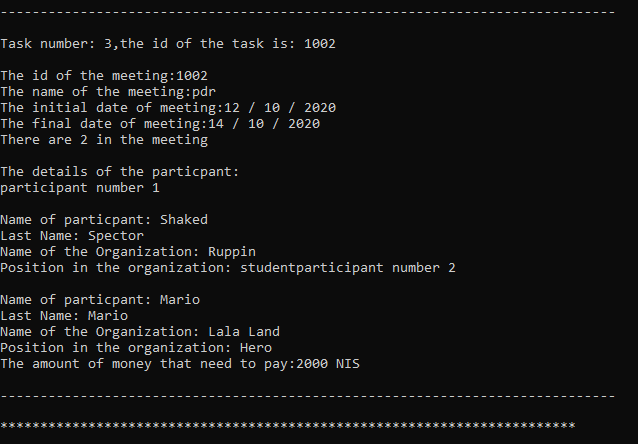
**תצלומי מסך מהDEMO אפשרות 1 בעליית המסך:**

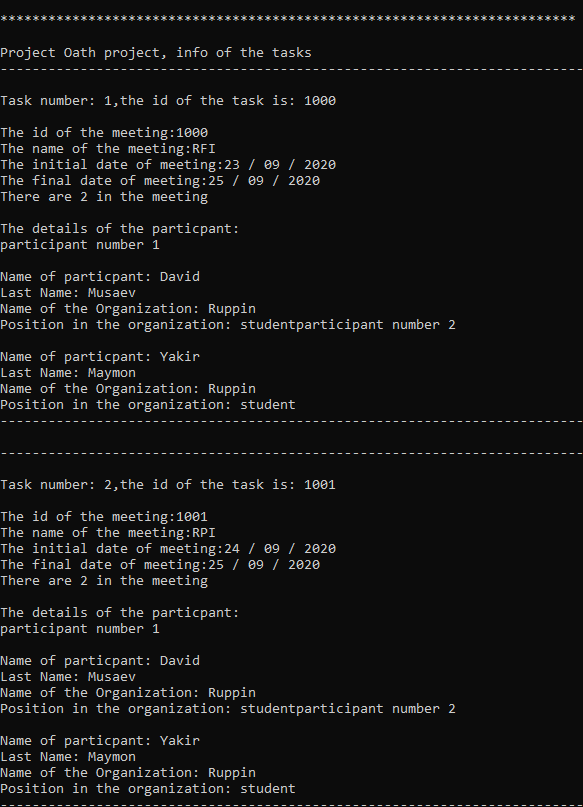


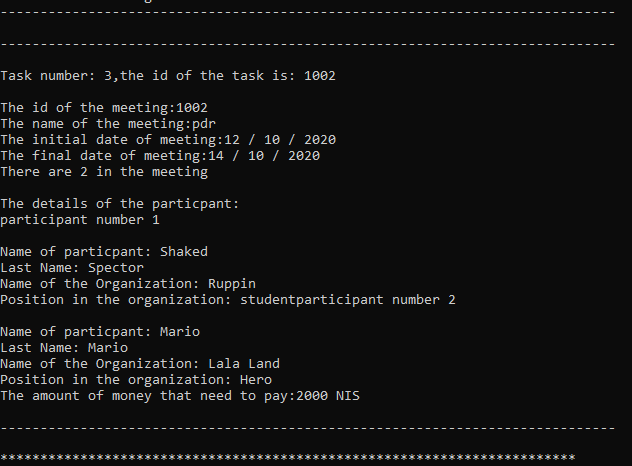


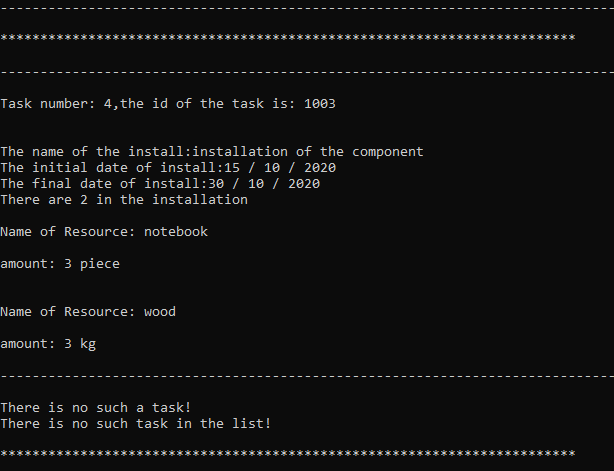


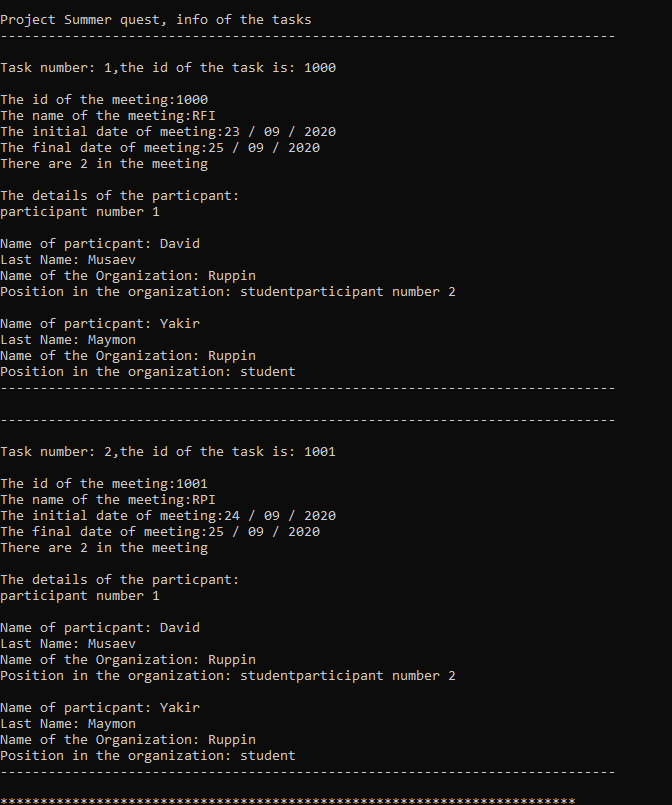


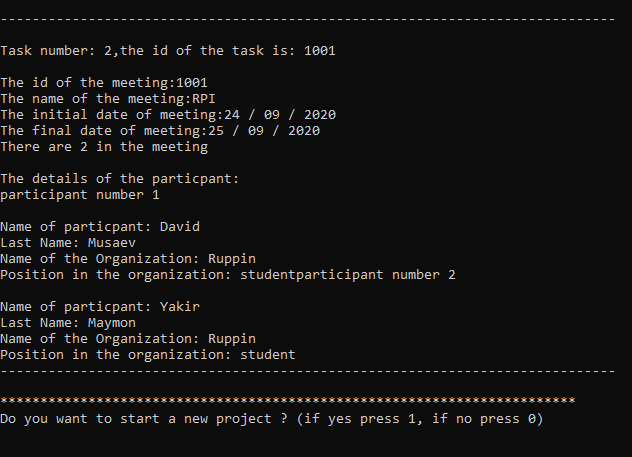




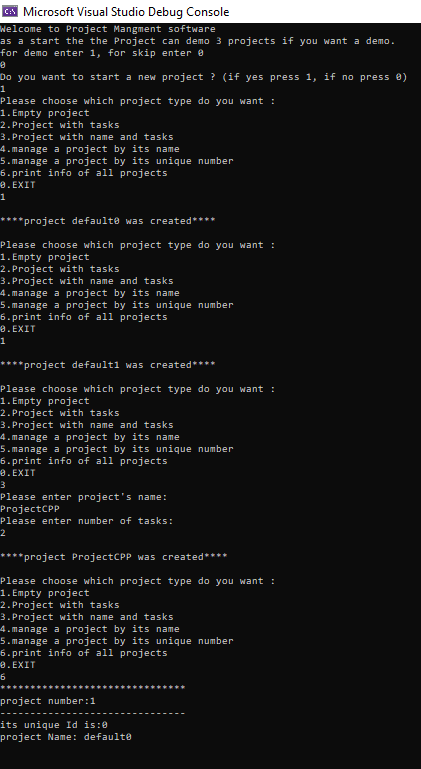








**תצלום מסך מתפריט למשתמש**



קטעי הקוד:

**Main.cpp:**

**Task.h:**

/\*task.h\*/

#ifndef STD\_LIB

#define STD\_LIB

#include<fstream>

#include<istream>

#include <iostream>

#include <string>

#include<ctime>

using namespace std;

#endif // !STD\_LIB

#define \_TASK\_H

/\*

class task

info about the class:task{this class contain information about the start of the task ,the end of the task and the name of the task}

abstract class: Y

derived class: N

child class of:

privet parameters in the class:

string initDate - start of the task

string finalDate - end of the task

string name - name of the task

const int numberTask - the number of the task

static int taskCt - how many tasks

\*/

class task

{

public:

//Constracturs

task(string initDate,string finalDate,string name);

task();

//Set Methods

void setName(string name)throw(string);/\*set the name of the task\*/

void setInitDate(string initDate)throw(string);//set initate date of a task

void setFinalDate(string finalDate)throw(string);//set final date of a task

//Get Methods

string getName()const { return name; };

string getInitDate()const { return initDate; };

string getFinalDate()const { return finalDate; };

int getNumberOfTask()const { return numberTask; };

static int getTaskCt(){ return taskCt; };

bool isExpiredA(string currentDate) const; /\*chek if the task final date is expired\*/

//Virtual Methods

virtual void PrintT(ostream& out)const = 0;/\*it is a virtual printing function,not working for this class\*/

//operators

/\*print operator\*/

friend ostream& operator<<(ostream& out, const task& ain)

{

out << "\nName of Task: "<<ain.getName();

return out;

}

/\*operator ==\*/

virtual bool operator==(const task& ain)

{

if (getInitDate() == ain.getInitDate() && getFinalDate() == ain.getFinalDate() && getName() == ain.getName())

return true;

else

false;

}

//Distractors

void removeT();

~task();

private:

string initDate; /\*start of the task\*/

string finalDate; /\*end of the task\*/

string name; /\*name of the task\*/

const int numberTask; /\* the number of the task\*/

static int taskCt; /\*how many tasks\*/

};

**Task.cpp:**

#include "task.h"

int task::taskCt = 1000;

task::task(string initDate, string finalDate, string name):numberTask(taskCt++)

{

setInitDate(initDate);

setFinalDate(finalDate);

setName(name);

}

task::task() :numberTask(taskCt++)

{

setInitDate("1/1/1900");

setFinalDate("30/1/1900");

setName("Pray for Peace");

}

/\*set the name of the task and checking if the name is ok\*/

void task::setName(string name)throw(string)

{

if (name == "") throw "Name is Empty String"; /\*chack if the name is empty\*/

if (&name == nullptr) throw "Name is Null"; /\*check if the name is null\*/

if (std::string::npos != name.find\_first\_of("0123456789"))

throw "Name Contains Digit";/\*check if there are digit in the name\*/

this->name = name;

}

/\*set initate date of a task and checking if the date is real\*/

void task::setInitDate(string initDate)

{

if (initDate == "") throw "Date is Empty "; /\*chack if the date is empty\*/

if (&initDate == nullptr) throw "Date is Null"; /\*check if the date is null\*/

if (std::string::npos != name.find\_first\_of("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz`!@#$%^&\*()\_\+="))

throw "Date Contains restedricted symbols";/\*check if there are letters or symbols in the date\*/

this->initDate = initDate;

}

/\*set final date of a task and chacking if the date is real\*/

void task::setFinalDate(string finalDate)

{

if (finalDate == "") throw "Date is Empty ";/\*chack if the date is empty\*/

if (&finalDate == nullptr) throw "Date is Null";/\*check if the date is null\*/

if (std::string::npos != name.find\_first\_of("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz`!@#$%^&\*()\_\+="))

throw "Date Contains restedricted symbols";/\*check if there are letters or symbols in the date\*/

this->finalDate = finalDate;

if (this->isExpiredA(initDate)) throw "final date expired";

}

/\*

reciving a string in format of "DD/MM/YY" end cheking if the object final date is expired,

return TRUE if expired FALSE id not expired

\*/

bool task::isExpiredA(string currentDate) const

{

string tmpFinal = this->getFinalDate(), tmpCurrent = currentDate;

string delimiter = " \ ", token0, token1;

size\_t pos0 = 0, pos1 = 0;

int day[2], month[2], year[2];

int i = 0;

while ((pos0 = tmpCurrent.find(delimiter)) != string::npos)//cut the date string and convert into 6 int

{

token0 = tmpCurrent.substr(0, pos0);

token1 = tmpFinal.substr(0, pos1);

if (i == 0)//first cut off the days

{

day[0] = stoi(token0);

day[1] = stoi(token1);

}

if (i == 1)//second cut off the month

{

month[0] = stoi(token0);

month[1] = stoi(token1);

}

if (i == 2)//third cut off the year

{

year[0] = stoi(token0);

year[1] = stoi(token1);

}

i++;

tmpCurrent.erase(0, pos0 + delimiter.length());

tmpFinal.erase(0, pos1 + delimiter.length());

}

if (year[0] > year[1])//if current year is higher then final

return true;//expired=true

else

{

if (month[0] > month[1])//if current month is higher then final month

return true;//expired=true

else

{

if (day[0] > day[1])//if current day is higher then final day

return true;//expired=true

}

}

return false;

return false;

}

/\*free all allocation\*/

void task::removeT()

{

delete this;

}

task::~task()

{

}

**Meeting.h:**

#ifndef \_TASK\_H

#define \_TASK\_H

#include "task.h"

#endif // !\_TASK\_H

#ifndef STD\_LIB

#define STD\_LIB

#include<fstream>

#include<istream>

#include <iostream>

#include <string>

using namespace std;

#endif // !STD\_LIB

#ifndef \_PART\_H

#define \_PART\_H

#include "participant.h"

#endif // !\_PART\_H

#define MAX\_PPL 5

#define \_MEETING\_H

/\*

class meeting

info about the class:meeting{this class contain information about the participants of the meeting and have the meeting loction }

abstract class: N

derived class: Y

child class of:task

privet parameters in the class:

int numberOfParticipants - number of participants

participant\* participantList[MAX\_PPL] - the list of all the participants in the meeting

string location - the loction of the meeting

\*/

class meeting:public task

{

public:

//Constracturs

meeting(string location, string initDate, string finalDate, string name, int numberOfParticipants=0);

meeting( string initDate, string finalDate, string name, int numberOfParticipants=0);

meeting(string location, string initDate, string finalDate, string name);

meeting();

//Set Methods

void setLocation(string location)throw(string); /\*set loction for the meeting\*/

void setNumberOfParticipants(int numberOfParticipants)throw(int); /\*set number of participants that will be in the meeting\*/

//Get Methods

string getLocation()const { return location; };

int getNumberOfParticipants()const { return numberOfParticipants; };

//Bool function return 1 if expired and if not it return 0

friend bool isExpired(string currentDate,string fDate);

/\*Operator +=\*/

meeting &operator+=(const participant &p);

/\*opertaor ==\*/

virtual bool operator==(const task& ing) const

{

if (((task\*)this)->operator==(ing) == false) {

return false;

}

const meeting \*meet = dynamic\_cast<const meeting\*>(&ing);

if (meet->getLocation() == this->getLocation() && meet->getNumberOfParticipants() == this->getNumberOfParticipants())

{

return true;

}

else return false;

}

//Print function

void PrintT(ostream& out)const;

//Distractors

void removeA();

~meeting();

private:

int numberOfParticipants; /\*number of participants\*/

participant \*participantList[MAX\_PPL]; /\*the list of all the participants in the meeting\*/

string location; /\*the loction of the meeting\*/

};

**Meeting.cpp:**

#include "meeting.h"

using namespace std;

meeting::meeting(string location, string initDate, string finalDate, string name, int numberOfParticipants):task(initDate,finalDate,name)

{

if (numberOfParticipants > MAX\_PPL)

{

numberOfParticipants = MAX\_PPL;

cout << "Max participants in meeting:" << MAX\_PPL <<"the value of participant has changed to max value"<< endl;

}

setNumberOfParticipants(numberOfParticipants);

this->setLocation(location);

}

meeting::meeting(string initDate, string finalDate, string name, int numberOfParticipants):task(initDate, finalDate, name)

{

this->setLocation("virtual:VC/ZOOM/Tel");

setNumberOfParticipants(numberOfParticipants);

int i = 0;

}

meeting::meeting(string location, string initDate, string finalDate, string name):task(initDate, finalDate, name)

{

int numberOfParticipants = 2;

setNumberOfParticipants(numberOfParticipants);

this->setLocation(location);

int i = 0;

}

meeting::meeting():task()

{

this->setLocation("home");

this->setNumberOfParticipants(0);

}

/\*set loction for the meeting and checking if the location name is not empty and not contain digits\*/

void meeting::setLocation(string location)throw(string)

{

if (location == "") throw "loction is Empty String"; /\*check is the location is not empty string\*/

if (&location == nullptr) throw "loction is Null"; /\*check if th location name is not null\*/

if (std::string::npos != location.find\_first\_of("0123456789")) /\*check if th location name does not contain digits\*/

throw "loction name Contains Digit"; /\*check if th location name does not contain digits\*/

this->location = location;

}

/\*set number of oarticipants in the meeting and check if the numbers is logical \*/

void meeting::setNumberOfParticipants(int numberOfParticipants)throw(int)

{

if (numberOfParticipants < 1)

throw numberOfParticipants; /\*check if the number of participant is logical \*/

string tmp;

this->numberOfParticipants = numberOfParticipants;

int i = 0;

cout << "For task number: " << this->getNumberOfTask() << "\nenter details about the participants.";

for (; i < numberOfParticipants; i++)

{

this->participantList[i] = new participant(); /\*alloction for participant\*/

cout << "\nEnter info of particpant number "<<i+1 <<"\nnenter the name of the participant: "; /\*enter participant first name\*/

cin >> tmp;

participantList[i]->setName(tmp);

cout << "enter the name of the last name of participant: "; /\*enter participant last name\*/

cin >> tmp;

participantList[i]->setLastName(tmp);

cout << "enter the name of the organization: "; /\*enter name of the organization\*/

cin >> tmp;

participantList[i]->setOrganization(tmp);

cout << "enter the position in the organization: "; /\*enter name of the position in the organization\*/

cin >> tmp;

participantList[i]->setPosition(tmp);

}

}

/\*?????\*/

meeting & meeting::operator+=(const participant & p)

{

if (this->getNumberOfParticipants() + 1 > MAX\_PPL)

cout << "Too many participants in the meeting" << endl;

else

{

this->participantList[getNumberOfParticipants() + 1]->setName(p.getName());

this->participantList[getNumberOfParticipants() + 1]->setLastName(p.getLastName());

this->participantList[getNumberOfParticipants() + 1]->setOrganization(p.getOrganization());

this->participantList[getNumberOfParticipants() + 1]->setPosition(p.getPosition());

}

return \*this;

}

/\*print the information of the meeting -

id,name,initial date,final date,participants details\*/

void meeting::PrintT(ostream& out)const

{

cout << "\nThe id of the meeting:" << this->getNumberOfTask(); /\*id of the meeting\*/

cout << "\nThe name of the meeting:" << this->getName(); /\*na,e of the meeting\*/

cout << "\nThe initial date of meeting:" << this->getInitDate(); /\*initial date of the meeting\*/

cout << "\nThe final date of meeting:" << this->getFinalDate(); /\*final date of the meeting\*/

cout << "\nThere are " << this->getNumberOfParticipants() << " in the meeting\n\nThe details of the particpant:"<<endl; /\*the participants details\*/

int i = 0;

for (; i < getNumberOfParticipants(); i++)

{

cout << "participant number " << i + 1 <<endl;

this->participantList[i]->printP();

}

}

/\*free allocation\*/

void meeting::removeA()

{

int i = 0;

for (; i < this->getNumberOfParticipants(); i++)

{

this->participantList[i]->removeP();

}

task::removeT();

}

/\*free all alloction\*/

meeting::~meeting()

{

this->removeA();

}

/\*checking if the date is expired - if current date == to final date tham the date int expired\*/

bool isExpired(string currentDate,string finalDate)

{

meeting \*tmp;

bool expired ;

tmp = new meeting();

tmp->setFinalDate (finalDate);

expired = tmp->isExpiredA(currentDate);

return expired;

}

**Install.h:**

#ifndef \_TASK\_H

#define \_TASK\_H

#include "task.h"

#endif // !\_TASK\_H

#ifndef \_RESOURCE\_H

#define \_RESOURCE\_H

#include "resource.h"

#endif // ! \_RESOURCE\_H

#ifndef \_MEETING\_H

#define \_MEETING\_H

#include "MEETING.h"

#endif // !\_MEETING\_H

#ifndef STD\_LIB

#define STD\_LIB

#include <iostream>

#include <string>

using namespace std;

#endif // !STD\_LIB

#define MAX\_AMOUNT\_RESOURCE 10

#define \_INSTALL\_H

/\*

class install

info about the class:install{this class contain information about what needed for instal the object - how many and which type of resource needed }

abstract class: N

derived class: Y

child class of:task

privet parameters in the class:

bool testInclude - test

int numberOfResource - number of resource

resource\* resourceList[MAX\_AMOUNT\_RESOURCE] - Resource reference

\*/

class install :public task

{

public:

//Constracturs

install(int numberOfResource, bool test, string initDate, string finalDate, string name);

install(bool test, string initDate, string finalDate, string name);

install(int numberOfResource, string initDate, string finalDate, string name);

install();

//Set Methods

void setNumberOfResource(int numberOfResource)throw(int); /\*set number of the resources\*/

void setTestInclude(bool testInclude)throw(string); /\*if in the installation included also test \*/

//Get Methods

bool getTestInclude() const { return testInclude; };

int getNumberOfResource()const { return numberOfResource; };

//Operators +=

install& operator+=(const resource& r);

//Opertator ==

virtual bool operator==(const task& ing) const

{

if (((task\*)this)->operator==(ing) == false) {

return false;

}

const install\* meet = dynamic\_cast<const install\*>(&ing);

if (meet->getTestInclude() == this->getTestInclude() && meet->getNumberOfResource() == this->getNumberOfResource())

{

return true;

}

else return false;

}

//Virtual Methods

virtual void PrintT(ostream& out)const; /\*it is a virtual printing function, not working for this class\*/

//Distractors

void removeI();

~install();

private:

bool testInclude; /\*test\*/

int numberOfResource; /\*number of resource\*/

resource\* resourceList[MAX\_AMOUNT\_RESOURCE]; /\*Resource reference\*/

};

**Install.cpp:**

#include "install.h"

install::install(int numberOfResource, bool test, string initDate, string finalDate, string name):task(initDate, finalDate, name)

{

this->setNumberOfResource(numberOfResource);

this->setTestInclude(test);

}

install::install(bool test, string initDate, string finalDate, string name) :task(initDate, finalDate, name)

{

this->setNumberOfResource(0);

this->setTestInclude(test);

}

install::install(int numberOfResource, string initDate, string finalDate, string name) :task(initDate, finalDate, name)

{

this->setNumberOfResource(numberOfResource);

this->setTestInclude(0);

}

install::install():task()

{

this->setNumberOfResource(0);

this->setTestInclude(0);

}

/\*

set number of the resources

and check if th number of resources is logic

\*/

void install::setNumberOfResource(int numberOfResource=0)thro w(int)

{

if (numberOfResource < 1)

throw numberOfResource;

string tmp;

double tmp1 = 0.0;

cout << "For task number: " << this->getNumberOfTask() << "\nenter details about the resources."; /\*enter details about the resources\*/

this->numberOfResource = numberOfResource;

int i = 0;

for (; i < numberOfResource; i++)

{

this->resourceList[i] = new resource;

cout << "\nEnter details of resource number "<<i+1<<"\nEnter Name of Resource: "; /\*enter name of resources\*/

cin >> tmp;

this->resourceList[i]->setNameOfResource(tmp);

cout << "\nEnter unit of mesure: ";

cin >> tmp;

this->resourceList[i]->setUnit(tmp);

cout << "\nEnter amount of unit of the resource: ";

cin >> tmp1;

this->resourceList[i]->setAmountOfResource(tmp1);

}

}

/\*if in the installation included also test \*/

void install::setTestInclude(bool testInclude)throw(string)

{

if (testInclude != 0 && testInclude != 1)throw("Bad bool value"); /\*check if the value is correct\*/

this->testInclude = testInclude;

}

/\*?????\*/

install & install::operator+=(const resource & r)

{

if(this->getNumberOfResource()+1>MAX\_AMOUNT\_RESOURCE)

cout << "Too many Resource in the install" << endl;

else

{

this->resourceList[getNumberOfResource() + 1]->setAmountOfResource(r.getAmountOfResource());

this->resourceList[getNumberOfResource() + 1]->setNameOfResource(r.getNameOfResource());

this->resourceList[getNumberOfResource() + 1]->setUnit(r.getUnit());

}

return \*this;

}

/\*print the information of the install -

name of the install,initial date,final date,number of resources\*/

void install::PrintT(ostream& out)const

{

cout << "\n\nThe name of the install:" << this->getName();

cout << "\nThe initial date of install:" << this->getInitDate();

cout << "\nThe final date of install:" << this->getFinalDate();

cout << "\nThere are " << this->getNumberOfResource() << " in the installation";

int i = 0;

for (; i < getNumberOfResource(); i++)

{

this->resourceList[i]->printR();

}

}

/\*free alloction\*/

void install::removeI()

{

int i = 0;

for (; i < this->getNumberOfResource(); i++)

{

this->resourceList[i]->removeR();

}

task::removeT();

}

install::~install()

{

removeI();

}

**Payment.h:**

#ifndef \_MEETING\_H

#define \_MEETING\_H

#include "meeting.h"

#endif // !\_MEETING\_H

#ifndef STD\_LIB

#define STD\_LIB

#include<fstream>

#include<istream>

#include <iostream>

#include <string>

using namespace std;

#endif // !STD\_LIB

#define \_PAYMENT\_H

/\*

class payment

info about the class:payment{this class contain information about a ameeting that in that neeting it needed to get pay}

abstract class: N

derived class: Y

child class of: meetinng and task

privet parameters in the class:

int amountOfCurrency - how much money needed

string currency - the type of the currency

\*/

class payment:public meeting

{

public:

//Constracturs

<<<<<<< HEAD

payment(string currency, string location, int numberOfParticipants, string initDate, string finalDate, string name, int amountOfCurrency = 1);

=======

payment( string currency, string location, int numberOfParticipants, string initDate, string finalDate, string name, int amountOfCurrency = 1);

>>>>>>> 5fd974218d193c097b82f44d48a4e7fce5aecc9e

payment();

//Set Methods

<<<<<<< HEAD

void setAmountOfCurrency(int amountOfCurrency )throw(int);/\*set how much money needed\*/

void setCurrency(string currency); /\*set the type of the currency\*/

=======

void setAmountOfCurrency(int amountOfCurrency )throw(int);

void setCurrency(string currency);

>>>>>>> 5fd974218d193c097b82f44d48a4e7fce5aecc9e

//Get Methods

int getAmountOfCurrency()const { return amountOfCurrency; };

string getCurrency()const { return currency; };

//Virtual Method

<<<<<<< HEAD

virtual void PrintT(ostream& out)const; /\*it is a virtual printing function, not working for this class\*/

=======

virtual void PrintT(ostream& out)const;

>>>>>>> 5fd974218d193c097b82f44d48a4e7fce5aecc9e

//Distractors

void removePA();

~payment();

private:

<<<<<<< HEAD

int amountOfCurrency; /\*how much money needed\*/

string currency;/\*the type of the currency\*/

=======

int amountOfCurrency;

string currency;

>>>>>>> 5fd974218d193c097b82f44d48a4e7fce5aecc9e

};

**Payment.cpp:**

#include "payment.h"

payment::payment(string currency, string location, int numberOfParticipants, string initDate, string finalDate, string name, int amountOfCurrency):meeting( location, initDate, finalDate, name, numberOfParticipants)

{

this->setCurrency(currency);

this->setAmountOfCurrency(amountOfCurrency);

}

payment::payment():meeting()

{

this->setCurrency("NIS");

this->setAmountOfCurrency(1);

}

/\*set how much money needed and check if the payment is ok\*/

void payment::setAmountOfCurrency(int amountOfCurrency)

{

if (amountOfCurrency < 0)

throw amountOfCurrency;/\*checking if the amount of currency is logic or real\*/

this->amountOfCurrency = amountOfCurrency;

}

/\*set the type of the currency\*/

void payment::setCurrency(string currency)

{

this->currency = currency;

}

/\*print the information of the meeting - amount of money needed to pay\*/

void payment::PrintT(ostream & out) const

{

meeting::PrintT(cout);

cout <<"\nThe amount of money that need to pay:"<<getAmountOfCurrency() <<" "<< getCurrency()<<endl;

}

/\*free meeting allocation\*/

void payment::removePA()

{

meeting::removeA();

}

/\*free payment alloction\*/

payment::~payment()

{

this->removePA();

}

**Participants.h:**

#ifndef STD\_LIB

#define STD\_LIB

#include <iostream>

#include <string>

using namespace std;

#endif // !STD\_LIB

#define \_PART\_H

/\*

class participant

info about the class:participant{this class contain information about the participant in the meetings}

abstract class: N

derived class: Y

privet parameters in the class:

string nameP - name of the participant

string lastName - last name of the participant

string organization - name of the organization

string position - the participant position int the organization

\*/

class participant

{

public:

//Constracturs

participant(string nameP,string lastName ,string organization,string position);

participant();

//Set Methods

void setName(string nameP)throw(string); /\*set the name of the participant \*/

void setLastName(string lastName)throw(string);/\* set the last name of the participant\*/

void setOrganization(string organization)throw(string); /\* set the name of the organization\*/

void setPosition(string poistion)throw(string); /\* set the participant position int the organization\*/

//Get Methods

string getName()const { return nameP; };

string getLastName()const{ return lastName; };

string getOrganization()const { return organization; };

string getPosition()const { return position; };

//Print function

void printP()const;

//Distractors

void removeP();

~participant();

private:

string nameP; /\*name of the participant\*/

string lastName; /\*last name of the participant\*/

string organization;/\*name of the organization\*/

string position;/\*the participant position int the organization\*/

};

**Participants.cpp:**

#include "participant.h"

participant::participant(string nameP, string lastName, string organization, string position)

{

setName(nameP);

setLastName(lastName);

setOrganization(organization);

setPosition(position);

}

participant::participant()

{

setName("Mario");

setLastName("Mario");

setOrganization("Lala Land");

setPosition("Hero");

}

/\* set name of the participant and check if the name is logic\*/

void participant::setName(string nameP)throw(string)

{

if (nameP == "") throw "Name is Empty String"; /\*check if the name is empty\*/

if (&nameP == nullptr) throw "Name is Null"; /\*check if the name is null\*/

if (std::string::npos != nameP.find\_first\_of("0123456789")) /\*check if the name contain digits\*/

throw "Name Contains Digit";

this->nameP = nameP;

}

/\*set last name of the participant and check if the name is logic\*/

void participant::setLastName(string lastName)throw(string)

{

if (lastName == "") throw "last Name is Empty String";/\*check if the last name is empty\*/

if (&lastName == nullptr) throw "last Name is Null";/\*check if the last name is null\*/

if (std::string::npos != lastName.find\_first\_of("0123456789"))/\*check if the last name contain digits\*/

throw "last Name Contains Digit";

this->lastName = lastName;

}

/\* set name of the Organization and check if the name is logic\*/

void participant::setOrganization(string organization)throw(string)

{

if (organization == "") throw "name of the organization is Empty String";/\*check if the Organization name is empty\*/

if (&organization == nullptr) throw "Name of the organization is Null";/\*check if the Organization name is null\*/

/\*the Organization name can contain digits\*/

this->organization = organization;

}

/\* set participant position int the Organization and check if the name is logic\*/

void participant::setPosition(string position)throw(string)

{

if (position == "") throw "name of the postion is Empty String";/\*check if the postion is empty\*/

if (&position == nullptr) throw "Name of the postion is Null"; /\*check if the postion is null\*/

this->position = position;

}

/\*print the information of the participant -

name of the participant,last name,organiztion name,Position in the organization\*/

void participant::printP()const

{

cout << "\nName of particpant: " << this->getName();

cout << "\nLast Name: " << this->getLastName();

cout << "\nName of the Organization: " << this->getOrganization();

cout << "\nPosition in the organization: " << this->getPosition();

}

/\*free alloction\*/

void participant::removeP()

{

delete this;

}

participant::~participant()

{

this->removeP();

}

**Resource.h:**

#ifndef STD\_LIB

#define STD\_LIB

#include<fstream>

#include<istream>

#include <iostream>

#include <string>

using namespace std;

#endif // !STD\_LIB

#define \_RESOURCE\_H

/\*

class resource

info about the class:resource{this class contain data of all the resources needed to install each project}

abstract class: N

derived class: N

privet parameters in the class:

double amountOfResource - amount of resource

string nameOfResource - the name of the resource

string unit - unit name

\*/

class resource

{

public:

//Constracturs

resource(string nameOfResource,string unit,double amountOfResource);

resource();

//Set Methods

void setNameOfResource(string nameOfResource)throw(string); /\*set the name of the resource\*/

void setUnit(string unit)throw(string); /\*set the name of the unit \*/

void setAmountOfResource(double amountOfResource)throw(int); /\* set the amount of resources\*/

//Get Methods

string getNameOfResource()const { return nameOfResource;};

double getAmountOfResource()const { return amountOfResource; };

string getUnit()const {return unit;};

//Print function

void printR()const;

//Distractors

void removeR();

~resource();

private:

double amountOfResource; /\*amount of resource\*/

string nameOfResource; /\*the name of the resource\*/

string unit; /\*unit name\*/

};

**Resource.cpp:**

#include "resource.h"

resource::resource(string nameOfResource, string unit, double amountOfResource)

{

setNameOfResource(nameOfResource);

setUnit(unit);

setAmountOfResource(amountOfResource);

}

resource::resource()

{

setNameOfResource("Tech");

setUnit("hour");

setAmountOfResource(1.0);

}

/\*set the name of the resource,check if the name is logic \*/

void resource::setNameOfResource(string nameOfResource)throw(string)

{

if (nameOfResource == "") throw "Name is Empty String";/\*check if the name is empty\*/

if (&nameOfResource == nullptr) throw "Name is Null";/\*check if the name is null\*/

if (std::string::npos != nameOfResource.find\_first\_of("0123456789"))/\*check if the name contain digits\*/

throw "Name Contains Digit";

this->nameOfResource = nameOfResource;

}

/\* set the amount of resources,check if the name is logic \*/

void resource::setAmountOfResource(double amountOfResource)throw(int)

{

if (amountOfResource < 0)

throw amountOfResource; /\*checking if the amount of resource is logic or real \*/

this->amountOfResource = amountOfResource;

}

/\*set the name of the unit, check if the name of the unit is real and logic \*/

void resource::setUnit(string unit)throw(string)

{

if (unit == "") throw "Unit name is Empty String";/\*check if the name is empty\*/

if (&unit == nullptr) throw "Unit name is Null";/\*check if the name is null\*/

this->unit = unit;

}

/\*print the information of the resource -

name of the resource,amount of resources\*/

void resource::printR() const

{

cout << "\n\nName of Resource: " << this->getNameOfResource();

cout << "\n\namount: " << this->getAmountOfResource()<<" " <<this->getUnit()<<endl;

}

/\*free resource allocation\*/

void resource::removeR()

{

delete this;

}

resource::~resource()

{

}

**Project.h:**

#ifndef \_MEETING\_H

#define \_MEETING\_H

#include "meeting.h"

#endif // !\_MEETING\_H

#ifndef \_INSTALL\_H

#define \_INSTALL\_H

#include "install.h"

#endif // !\_INSTALL\_H

#ifndef \_PAYMENT\_H

#define \_PAYMENT\_H

#include "payment.h"

#endif // !\_PAYMENT\_H

#ifndef STD\_LIB

#define STD\_LIB

#include<fstream>

#include<istream>

#include <iostream>

#include <string>

using namespace std;

#endif // !STD\_LIB

#define \_PROJECT\_H

/\*

class project

info about the class:project

{this class can Bind into it a set of tasks according to the participant's input

and actually allow the user access to functions that give an overall view of the project.}

abstract class: N

derived class: N

privet parameters in the class:

string projectName - the name of the project

int totalProjectTask - total tasks in the project

task\*\* taskList - task list

int indexTask - index of each task

\*/

class project

{

public:

project();

project(int totalProjectTask,string projectName);

project(int totalProjectTask);

void setTotalProjectTask(int totalProjectTask)throw(int); /\*set the project total tasks\*/

int getTotalProjectTask()const { return totalProjectTask; };

void setProjectTasks(int totalProjectTask); /\*set the project tasks\*/

void setIndexTask()throw(int); /\*set the index of each task\*/

int getIndexTask()const { return indexTask; };

void setProjectName(string projectName)throw(string); /\*set the project name\*/

string getProjectName()const { return projectName; };

int searchlist(int taskNumber); /\*search if the task is at the task list\*/

void printInfo()throw(string); /\*this function print the information of the project\*/

void printInfo(int i); /\*print the index of each task\*/

/\*operator +=\*/

void operator+=(task& other)

{

this->setIndexTask();

if (this->totalProjectTask > this->getIndexTask())

{

this->taskList[this->getIndexTask()] = &other;

}

}

/\*operator -=\*/

void operator-=(int other)

{

if (this->taskList[other])

{

delete this->taskList[other];

this->taskList[other] = NULL;

setIndexTask();

}

}

//Distractor

~project();

private:

string projectName;/\*the name of the project\*/

int totalProjectTask; /\*total tasks in the project\*/

task \*\*taskList; /\*task list\*/

int indexTask; /\*index of each task\*/

};

**Project.cpp:**

#include "project.h"

project::project()

{

this->setProjectName("default");

this->setIndexTask();

this->setTotalProjectTask(0);

this->setProjectTasks(0);

}

project::project(int totalProjectTask,string projectName)

{

this->setProjectName(projectName);

this->setIndexTask();

this->setTotalProjectTask(totalProjectTask);

this->setProjectTasks(totalProjectTask);

}

{

this->setProjectName("default");

this->setProjectName(projectName);

this->setIndexTask();

this->setTotalProjectTask(totalProjectTask);

this->setProjectTasks(totalProjectTask);

}

/\*set the project total tasks,check if the total check is logic\*/

void project::setTotalProjectTask(int totalProjectTask)throw(int)

{

if (totalProjectTask < 0)

throw totalProjectTask;/\*check if the total check is logic\*/

if (totalProjectTask)

this->totalProjectTask = totalProjectTask;

}

/\*set the project tasks\*/

void project::setProjectTasks(int totalProjectTask)

{

if (totalProjectTask)

{

this->taskList = new task \*[totalProjectTask];

int i = 0;

for (; i < totalProjectTask; i++)

{

this->taskList[i] = NULL;

}

}

else this->taskList = NULL;

}

/\*set the index of each task\*/

void project::setIndexTask()throw(int)

{

int i = 0;

for (i = 0; i < this->totalProjectTask; i++)

{

if (this->taskList[i] == NULL)

{

this->indexTask = i;

return;

}

if((indexTask<0)||(indexTask>totalProjectTask))throw(indexTask);

}

}

/\*set the project name, check if the name of the project is logic\*/

void project::setProjectName(string projectName)throw(string)

{

if (projectName == "") throw "Name of the project is Empty String"; /\*check if the name is empty\*/

if (&projectName == nullptr) throw "Name of the project is Null"; /\*check if the name is null\*/

this->projectName = projectName;

}

/\*search if the task is at the task list\*/

int project::searchlist(int taskNumber)

{

for (int i = 0; i < this->getIndexTask()+1; i++)

{

if (this->taskList[i] && taskNumber == this->taskList[i]->getNumberOfTask()) return i;

}

cout << "\nThere is no such a task!" << endl; /\*if there is no task like the user insert \*/

return -1;

}

/\*this function print the information of the project\*/

void project::printInfo()throw(string)

{

cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\nProject " << this->getProjectName() << ", info of the tasks";

for (int i = 0; i < this->getIndexTask(); i++)

{

if (this->taskList[i])

{

cout << "\n-----------------------------------------------------------------------------\n";

cout << "\nTask number: " << i + 1 << ",the id of the task is: " << this->taskList[i]->getNumberOfTask() << endl;

this->taskList[i]->PrintT(cout);

cout << "\n-----------------------------------------------------------------------------\n";

}

else

{

cout << "\nThere is no such a task!" << endl;

}

}

cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

}

/\*print the index of each task\*/

void project::printInfo(int i)

{

if (this->taskList[i] && i>=0)

{

cout << "\n-----------------------------------------------------------------------------\n";

cout << "\nTask number: " << i + 1 << ",the id of the task is: " << this->taskList[i]->getNumberOfTask() << endl;

this->taskList[i]->PrintT(cout);

cout << "\n-----------------------------------------------------------------------------\n";

}

}

/\*Distractor and free alloction\*/

project::~project()

{

for (int i = 0; i < this->getTotalProjectTask(); i++)

{

if(this->taskList[i])

this->taskList[i]->removeT();

}

delete[] this->taskList;

}