152114002: NESNE TABANLI PROGRAMLAMA I
2024-2025 GÜZ DÖNEMİ
DÖNEM PROJESİ (I. AŞAMA)
Son Teslim Tarihi: 22 Aralık 2024, Pazar, 23:59

Not: Projenin II. Aşaması 22 Aralık 2024 tarihinde verilecektir. II. Aşamanın teslim tarihi 29 Aralık 2024 olarak planlanmaktadır.

Açıklama ve Kurallar:

1. Grup çalışması içerisinde,
   - **i.** Bütün grup üyeleri, tasarım sürecinde bulunmalıdır.
   - **ii.** Görevler (sınıf kodlarının ve uygulamanın yazılması) tüm grup üyelerine dağıtılmalı ve raporda bu dağılım açıkça belirtilmelidir.
   - **iii.** Her öğrenci, kod sürüm takip sistemi olarak kullanılan bitbucket üzerinde hesap açarak (laboratuvarda nasıl kullanılacağı anlatılacaktır), her grupta bir takım lideri seçilecek, lider bir proje oluşturacak, takım üyeleri kendilerine atanan kaynak dosyalar üzerinde aynı projede çalışacaktır. Değerlendirmede, her bir takım üyesinin buradaki hareketi dikkate alınacaktır.
   - **iv.** Her öğrenci, kendisine atanan sınıf kodlarını ve test programlarını yazmalıdır. Her sınıf ayrı header (.h) ve ayrı source (.cpp) dosyasına sahip olmalıdır. Her dosyada kodu yazan kişinin ismi ve tarih bulunmalıdır.
2. Kodların yanında ilgili kod parçası ile ilgili açıklamaların yazılması gerekmektedir. (Bakınız EK A.1).
3. Proje için, tasarım ve gerçekleme aşamasının aktarıldığı bir rapor hazırlanması gerekmektedir. (Bakınız EK A.2)
4. Nesne tabanlı yapıların kullanıldığı (abstraction, inheritance, polymorphism, templates, exception handling, etc) iyi bir tasarım ve kodlama yapmalısınız.
5. Proje için, yazılan kodları ve proje raporunu sıkıştırarak. zip ya da .rar olarak, ve dosyayı "Grup_Uyelerinden_birinin_ismi.zip/rar" şeklinde isimlendirip, DYS yüklemelisiniz.
6. Notlar, ekte verilen tabloya göre verilecektir. (Bakınız EK A.3)
7. Uygulama programı konsol tabanlı olmalıdır.
8. Sadece standart C++ kütüphaneleri kullanılmalıdır.
9. **Tasarım ve/ve ya kodlarınızın kısmen ya da tamamen başka gruplardan ve referans gösterilmemiş kaynaklardan alındığı belirlenirse, sıfır not alırsınız.**

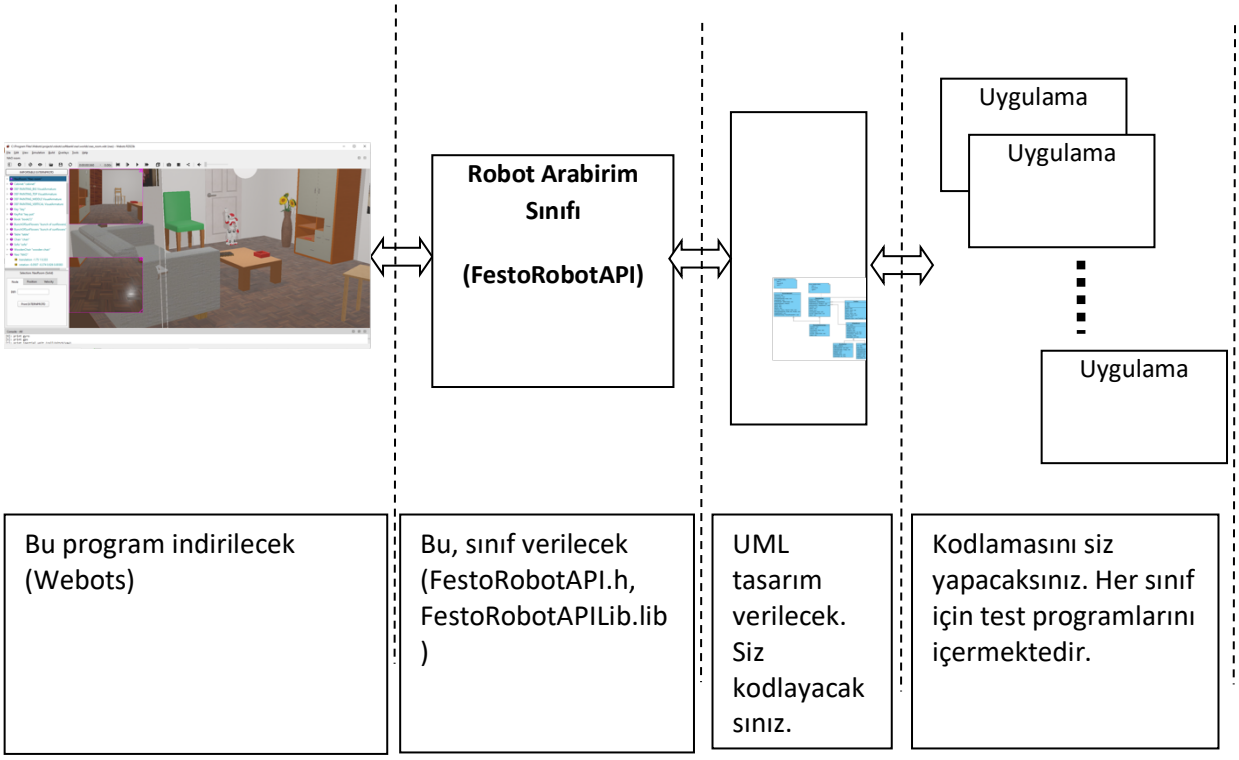
10. **Dönem sonunda, proje gruplarının sunum yapması istenebilir.**

PROJE BAŞLIĞI: Her Yöne Hareketli (Omni-directional) Robot Denetim Sistemi

Proje Açıklaması:

Proje konusu, 3B robot simülatöründeki robotun denetimi (hareket ettirilmesi ve sensörlerinden veri alınması) için yazılım geliştirmektedir. Bu projede, Webots adlı simulator programı ve bu programdaki robotu denetleyebilmek için gereken sınıf kütüphanesi verilmektedir.

Şekil 1' de görüldüğü gibi, Webots robot simülatörünü indirip kuracaksınız (Bknz. EK-1). Simülatör programında robot bulunmaktadır. Bu robot, yazılan programlarla hareket ettirilebilmektedir. Programa erişim, TCP/IP protokolü ile haberleşerek yapılmaktadır. Siz bu karmaşık programlama yapıları ile uğraşmayacaksınız. Bu karmaşık kodlamalar, FestoRobotAPI adlı sınıfın altında gerçekleştirilmektedir. Siz uygulama geliştirirken simülatördeki robota erişim için bu sınıfı kullanacaksınız. Bu sınıfı projenizde kullanabilmeniz ve projenizi derleyebilmeniz için (FestoRobotAPI.h, FestoRobotAPI.lib) dosyaları size verilmektedir. Ayrıca, FestoRobotAPI sınıfının üye fonksiyonlarının açıklamaları FestoRobotAPI.h dosyasında vardır. Ayrıca, size verilen RobotController.cpp programı da sınıfın kullanımını göstermek ve test etmek için size yardımcı olacaktır.

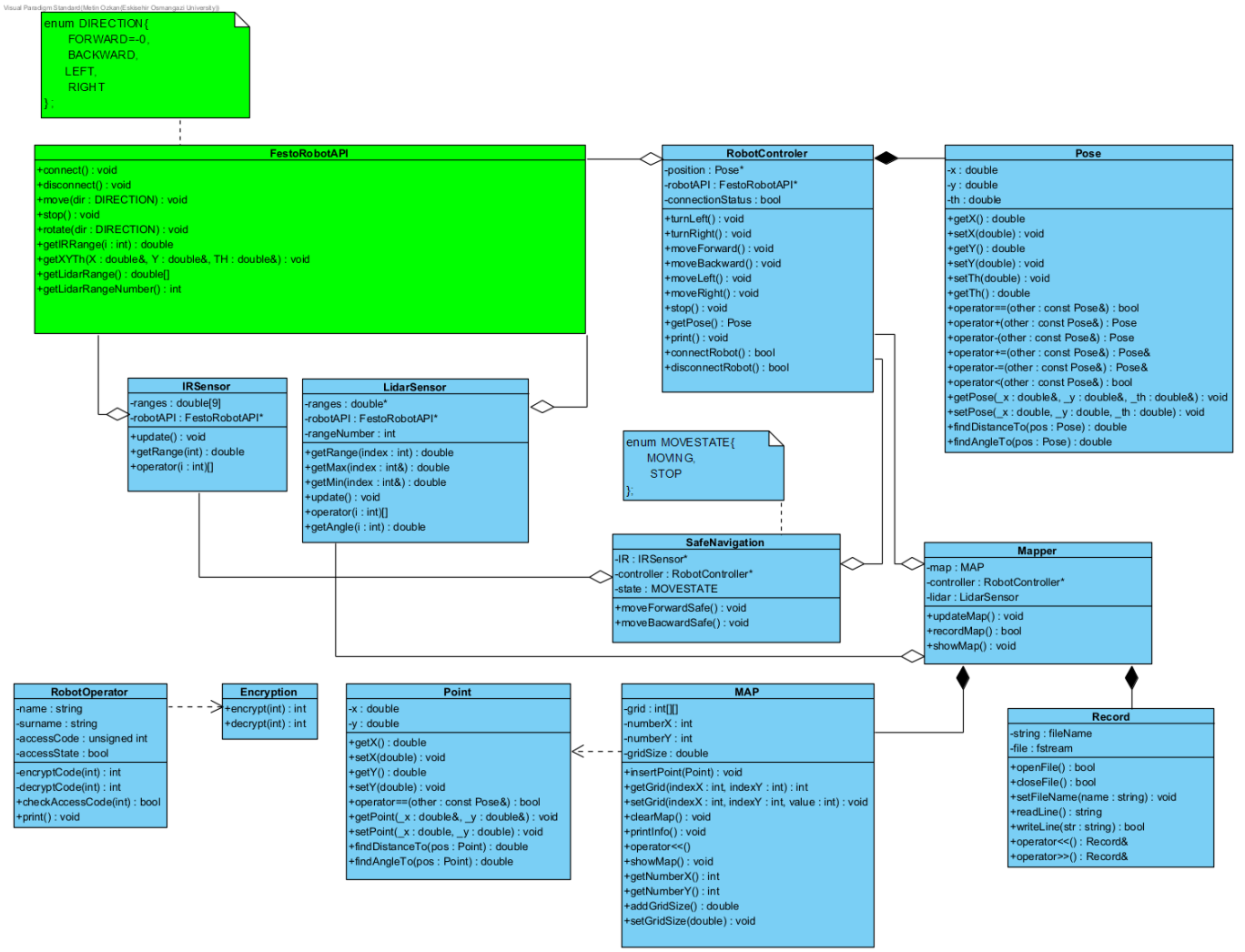


Şekil 1.

Projeye başlarken, öncelikle EK-1'de adım adım verilen işlemleri gerçekleştiriniz. Daha sonra sizden beklenen kodlamaları ve testleri gerçekleştiriniz.

Kısım 1.

Bu kısımda, Şekil 2'de verilen tasarımı kodlamanız beklenmektedir. Tasarımın UML gösteriminde, eksiklikler olabilir. Kodlama esnasında, tespit edeceğiniz bu eksikleri sizlerin gidermesi gerekiyor. Sınıflara, kodlama esnasında gereken üyeler eklenebilir.



Şekil 2.

Class FestoRobotAPI: Bu sınıf size simülatör ile kullanmanız ve simülatör ile çalışacak programları geliştirebilmeniz için veriliyor. Elinizde, kaynak kodu ".cpp" olmadığında değiştirme şansınız yoktur. Ancak, kullanacağınız üye fonksiyonların ne yaptığı ve gerektirdiği parametrelerin açıklamaları, FestoRobotAPI.h dosyasında bulunabilir.

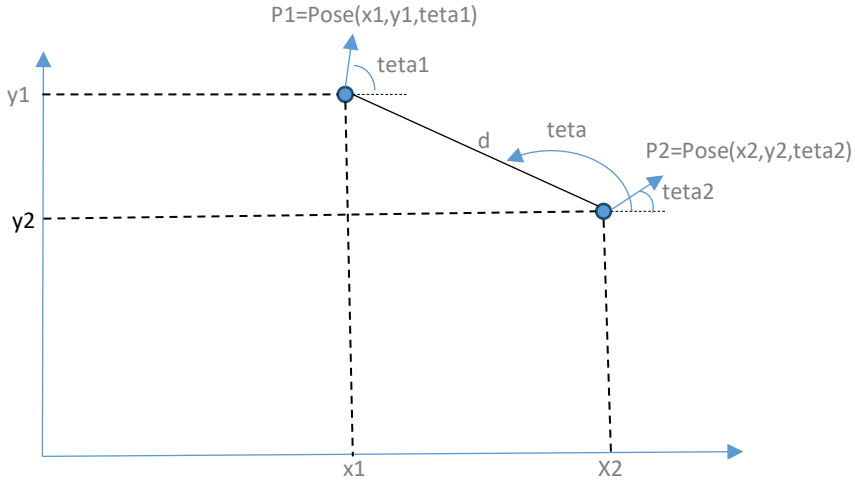
Class RobotControler: Bu sınıf, robotun hareket ettirilmesi ve robotun konumu ile ilgili bilgilere erişmeyi sağlar. Fonksiyonlarını, NaoRobotAPI sınıfındaki fonksiyonlardan faydalanarak gerçekler.

turnLeft(): Robotun sola dönmesini sağlar.
turnRight(): Robotun sağa dönmesini sağlar.
forward(): Robotun, sürekli ilerlemesini sağlar.
backward(): Robotun, sürekli geriye doğru ilerlemesini sağlar.
getPose(): Robotun, konumunu döndürür.
stop (): Robotun, durmasını sağlar.
moveLeft(): Robotun, sola doğru yanlamasına ilerlemesini sağlar.
moveRight(): Robotun, sağa doğru yanlamasına ilerlemesini sağlar.
print(): Robotun, bilgilerini ekrana bastırır.
connectRobot(): Robota bağlanır.
disconnectRobot(): Robota bağlantıyı keser.

Class Pose: Bu sınıf, robotun konum bilgilerini tutma ve yönetme görevine sahiptir. "th" robot yönelimini belirtmektedir. x ve y, metre biriminde değer alır. th ise, derece biriminde değer alır.

d=P1.findDistanceTo(P2); and d=P2.findDistanceTo(P1);

teta=P2.findAngleTo(P1);



Class IRSensor: IR mesafe sensörü için veri tutma ve yönetimini sağlar.

update (): Robota ait güncel sensör mesafe değerlerini, *ranges* dizisine yükler.

getRange(i): i. İndeksine sahip sensörün mesafe bilgisini döndürür.

operator[]: indeksi verilen sensör değerini döndürür. *getRange(i)* ile benzer fonksiyonu gerçekler.

Class LidarSensor: Lidar sensör verilerini yönetir.

update (): Robota ait güncel mesafe değerlerini, *ranges* dizisine yükler. Veri sayısı, *rangeNumber* üyesinde tutulur.

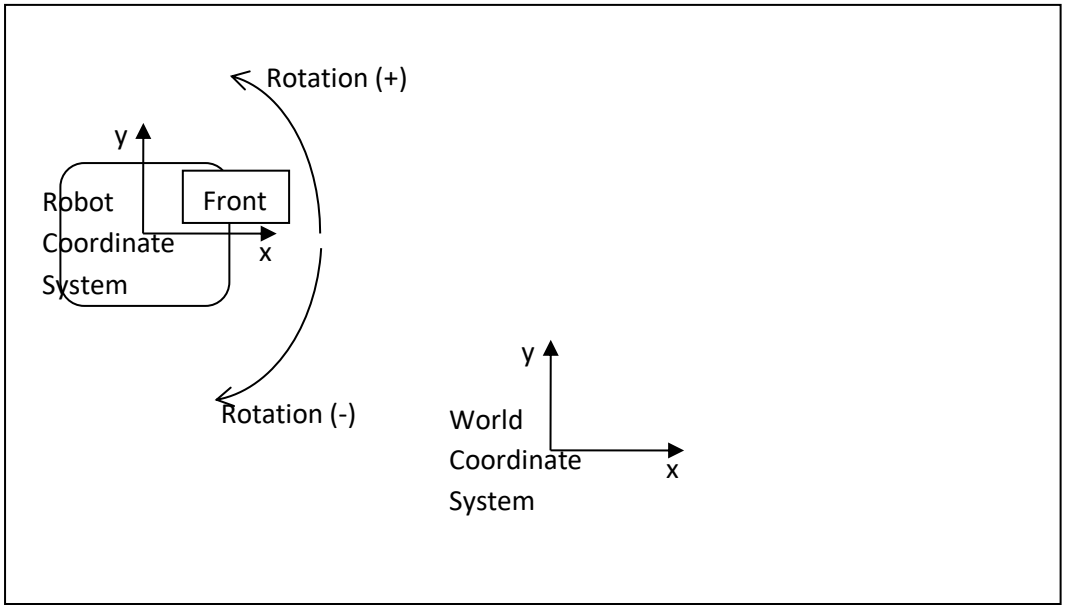
getRange(i): i. indeksine sahip lidar verisini döndürür (uzaklık, metre olarak).

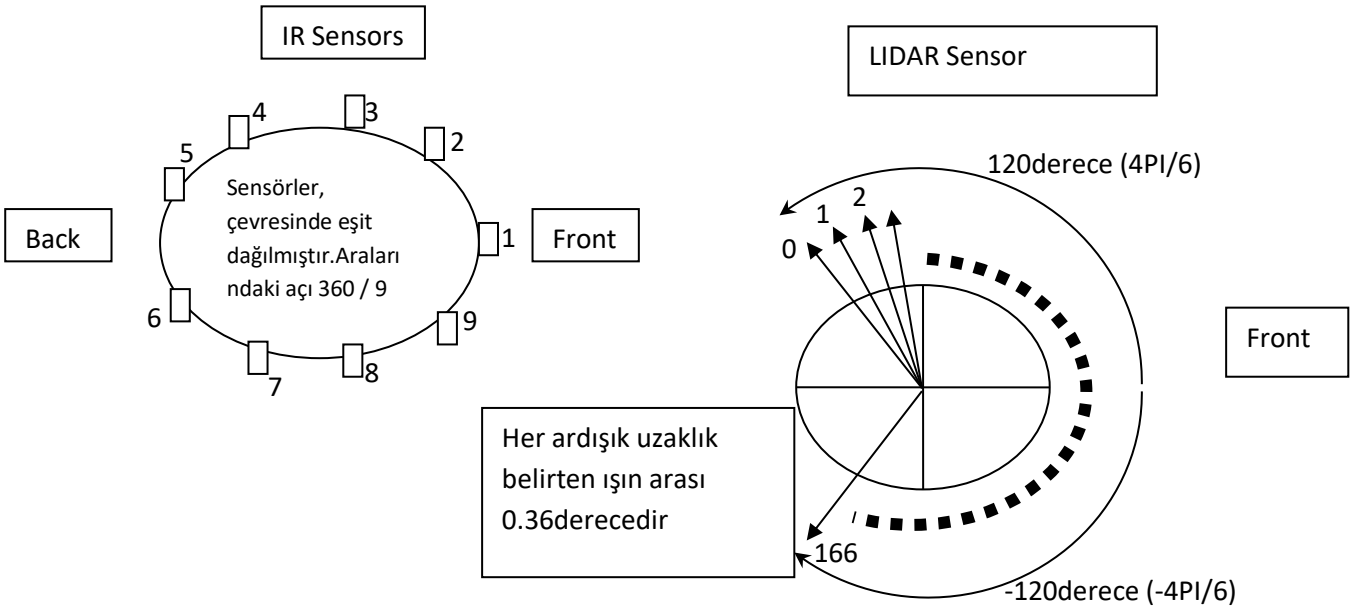
getMin(index): Bütün lidar verileri içinde en uzak mesafeye sahip verinin indeksini ve mesafeyi döndürür.

getMax(index): Bütün lidar verileri içinde en kısa mesafeye sahip verinin indeksini ve mesafeyi döndürür.

operator[]: indeksi verilen lidar uzaklığını döndürür. getRange(i) ile benzer fonksiyonu gerçekler.

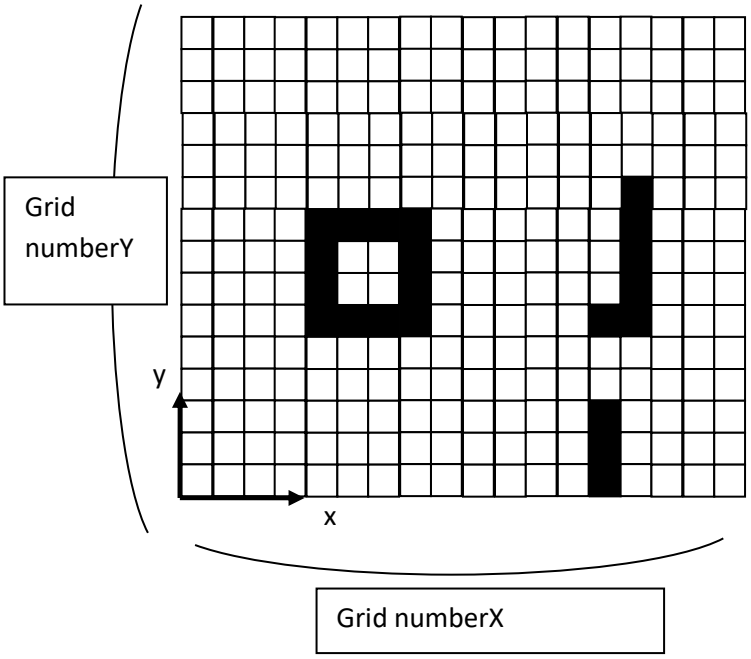
getAngle(i): i indeksindeki lidar verisinin robota göre açısını döndürür.





Class MAP: Robotun dolaştığı ortam aşağıda şekilde olduğu gibi eşit büyüklükte karelerde oluşan grid iye temsil edilir. Bu gridler iki boyutlu matris olarak tutulur. Boş olanlar, 0, dolu olanlar 1 değeri tutar. Başangıçta hepsi 0 değerine sahiptir.

insertPoint(Point): Noktanın gridler üzerinde denk geldiği kare saptanır ve o kare 1 değeri atılır.
getGrid() ve setGrid: Belirtilen indeksteki grid değerini döndürür ve atanır.
clearMap(): Tüm gridlere 0 değeri atılır.
printInfo(): Harita bilgileri ekranı bastırılır.
showMap(): Grid içeriklerini ekrana yazdırır. Boş olan gridlere "." Dolu olan gridlere "x" yazar.
getNumberX ve getNumberY: Gridlerin x ve y eksenlerinde grid sayılarını döndürür.
addGridSize() ve setGridSize(): Grid boyutlarını döndürür ve atar.



Gridler karelerden oluşur ve hepsi eşit boyuttadır. Beyaz olan gridler içi boş olan, siyah olanlar da bir engele denk geldiği için dolu olanlardır.

Class SafeNavigation: Bazı güvenli hareketleri yaptırır.

*moveForwardSafe():*Robot ileri gider. Bir engele 0.5m yaklaştığında durur.

*moveBackwardSafe():*Robot geri gider. Bir engele 0.5m yaklaştığında durur.

Class Mapper: Harita oluşturma ve yönetme.

*updateMap():*Lidar verilerini alır, her bir veriyi world koordinat sistemine dönüştürür. Hangi gride denk geldiğini bulup, o gridi dolu olarak işaretler.

*RecordMap():*Haritayı dosyaya kaydeder.

showMap(): Grid içeriklerini ekrana yazdırır. Boş olan gridlere "." Dolu olan gridlere "x" yazar.

Class Record: Robot uygulamalarında kullanılmak üzere, dosyaya kaydetme işlemlerini yürütür.

openFile(): Yazdırma ve okuma yapılacak dosya açar.

closeFile(): Yazdırma ve okuma yapılacak dosya kapatır.

setFileName(): Yazdırma ve okuma yapılacak dosya adını alır.

readLine(): Dosyadan bir satır veriyi okur.

writeLine(): Dosyaya bir satır veriyi yazar.

operator<<(): Verileri dosyaya yazar.

operator>>(): Verileri dosyadan alır.

Class RobotOperator: Robot uygulamalarında kullanılmak üzere, robotu kumanda edecek operatörün yetkilendirilmesinde kullanılacak bir sınıftır.

encryptCode(int): 4 rakamdan oluşan kodu, Encryption sınıfının fonksiyonu kullanılarak şifrelendirir.

decryptCode(int): 4 rakamdan oluşan kodu, Encryption sınıfının fonksiyonu kullanılarak şifresini çözdürür.

checkAccessCode(int): Girilen kodun, şifrelenmiş olarak tutulan accessCode ile aynı olup olmadığını kontrol eder.

print(): Operatorun, adını soyadını ve erişim durumunu ekrana getirir.

accessCode üye verisinin değeri, constructor fonksiyonu ile alınacak ve şifrelenmiş halini tutacaktır.

Class Encryption: Bu sınıf, 4 rakamlı sayıların şifrelenmesini ve şifre çözümünü yapmaktadır. Şifreleme ve şifre çözümü aşağıda açıklandığı algoritma ile gerçekleştirilmektedir.

Replace each digit with the result of adding 7 to the digit and getting the remainder after dividing the new value by 10. Then swap the first digit with the third, and swap the second digit with the fourth. Then print the encrypted integer. Write a separate application that inputs an encrypted four-digit integer and decrypts it (by reversing the encryption scheme) to form the original number.

Kısım 2.

Yukarıdaki sınıfların her birisi için bir test programı yazılacaktır. Bu program kaynak dosyaları, RobotControlerTest.cpp, IRSensorTest.cpp, vb. şeklinde adlandırılacaktır. Her bir program, ilgili sınıfı kodlayan grup üyesi tarafından yazılacaktır. Test programı, sınıfın her bir üye fonksiyonlarını, aykırı değerleri de içerecek şekilde kapsayacaktır. Test programında yaşanan beklenmeyen durumlar ve hatalara göre, sınıf kodları iyileştirilecektir.

Kısım 3.

Kısım 1 ve 2 tamamlandıktan sonra yazılan sınıfları kullanan bir menü tasarlayınız. Menüler, metin tabanlı olacaktır. Menüleri, nesne tabanlı yaklaşım ile tasarlayınız, UML diyagramını oluşturup, kodlayınız.

Menülerin kullanımına bir örnek:

```
Main Menu
1. Connection
2. Motion
3. Sensor
4. Quit
Choose one : 1 ↵

Connection Menu
1. Connect Robot
2. Disconnect Robot
3. Back
Choose one : 1 ↵

<Connect>
Robot is connected...

Connection Menu
1. Connect
2. Disconnect
3. Back
Choose one : 2 ↵

<Disconnect>
Robot is disconnected...

Connection Menu
1. Connect
2. Disconnect
3. Back
```

```
Choose one : 3 ↲

Main Menu
1. Connection
2. Motion
3. Sensor
4. Quit
Choose one : 2 ↲


Motion Menu
1. Move robot
2. Safe Move Robot
3. Turn left
4. Turn Right
5. Forward
6. Move distance
7. Close Wall
8. Quit
Choose one : 3 ↲

          .
          .
          .
```

Kısım 4.

Menülerin kullanılarak, robot denetiminin yapılabildiği bir uygulama programı yazınız. Test yaparak raporda sonuçları paylaşınız.

DOSYA EKLERİ:

EK 1. Geliştirme Ortamının Oluşturulması

EK 2. Festo robot denetimi için kütüphane (FestoRobotAPI.h, FestoRobotAPILib.lib)

EK 3. Festo robot denetimi için test programı kaynak dosyası (RobotController.cpp)

Appendix:

A.1 Doxygen HowTo

Doxygen is a documentation system for C++, C, Java, Objective-C, Python, IDL (Corba and Microsoft flavors), Fortran, VHDL, PHP, C#, and to some extent D.

It can help you in three ways:

1. It can generate an on-line documentation browser (in HTML) and/or an off-line reference manual (in LaTeX) from a set of documented source files. There is also support for generating output in RTF (MS-Word), PostScript, hyperlinked PDF, compressed HTML, and Unix man pages. The documentation is extracted directly from the sources, which makes it much easier to keep the documentation consistent with the source code.
2. You can configure doxygen to extract the code structure from undocumented source files. This is very useful to quickly find your way in large source distributions. You can also visualize the relations between the various elements by means of include dependency graphs, inheritance diagrams, and collaboration diagrams, which are all generated automatically.
3. You can also use doxygen for creating normal documentation (as I did for this manual).

Doxygen is developed under Linux and Mac OS X, but is set-up to be highly portable. As a result, it runs on most other Unix flavors as well. Furthermore, executables for Windows are available.

For more detail, http://www.stack.nl/~dimitri/doxygen/manual.html

Sample :

The Header File (TestA.h) is:

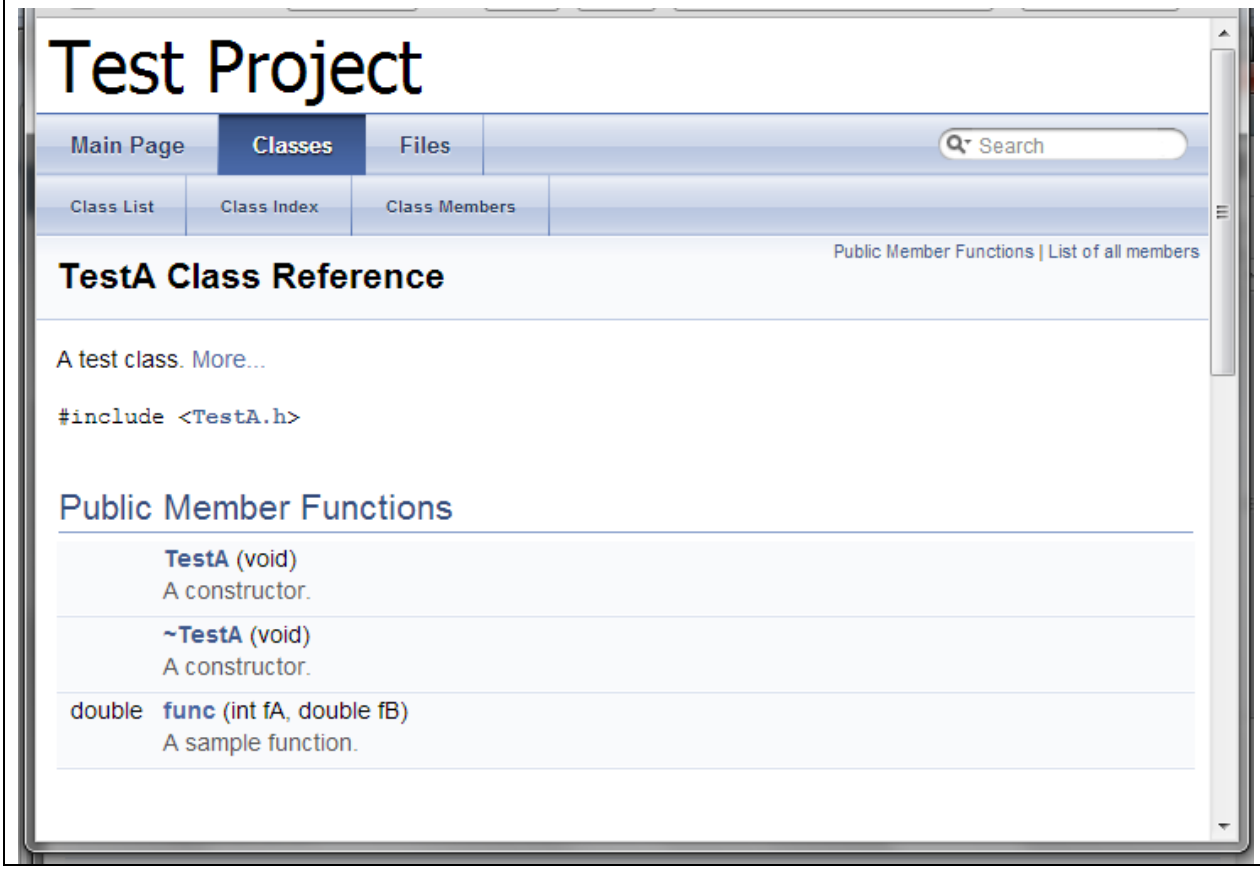
```
/**
 * @file   TestA.h
 * @Author Me (me@example.com)
 * @date   September, 2008
 * @brief  Brief description of file.
 *
 * Detailed description of file.
 */

//! An enum.
/*! More detailed enum description. */
enum TestENUM{
     ENUM1,  /*!< Definition of ENUM1 */
     ENUM2,    /*!< Definition of ENUM2 */
     ENUM3     /*!< Definition of ENUM3 */
};

//!  A test class.
/*!
  A more elaborate class description.
*/
class TestA{
public:
     //! A constructor.
     TestA(void);
     //! A constructor.
```

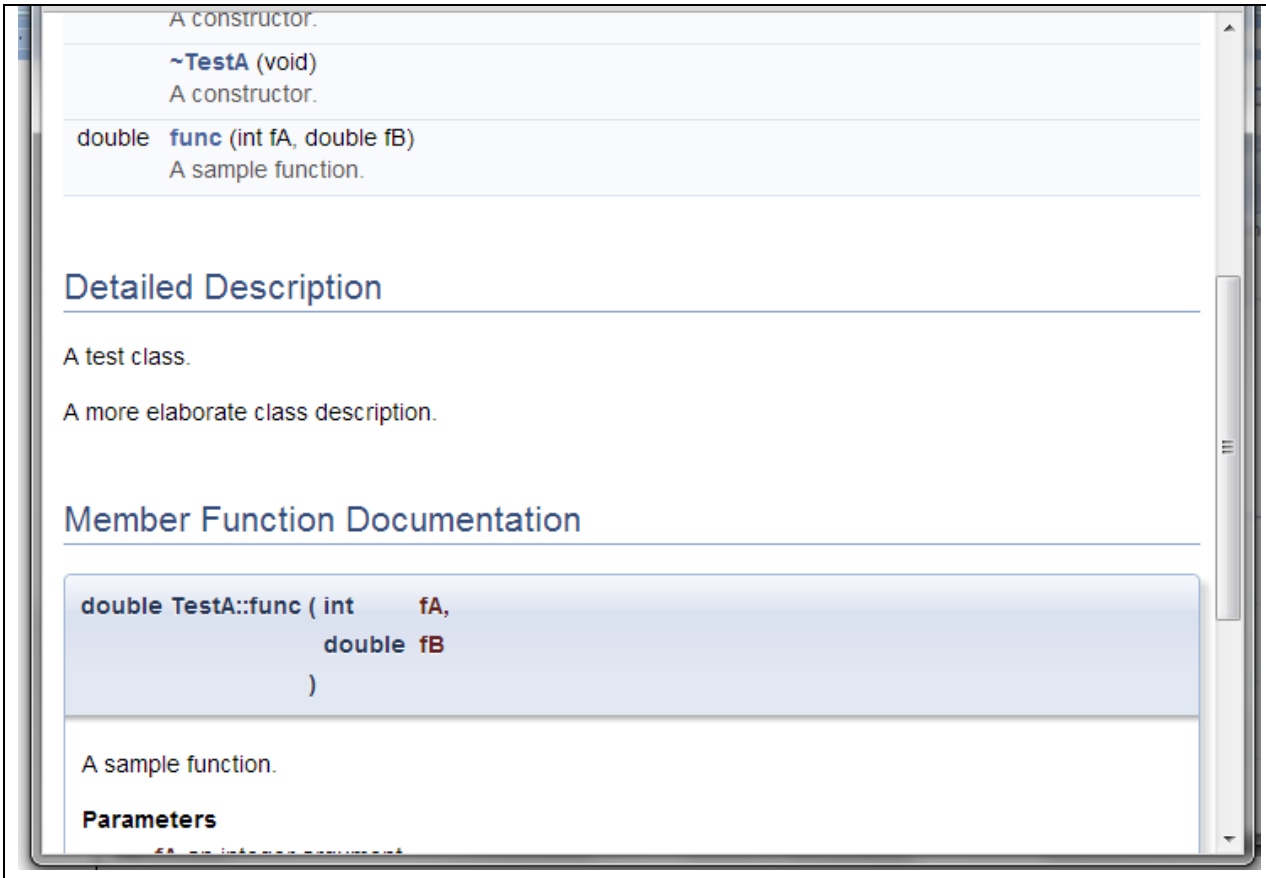
```
      ~TestA(void);
      //! A sample function.
      double func(int fA, double fB);
};
```

The Source File (TestA.cpp) is:

```
#include "TestA.h"

TestA::TestA(void)
{}

TestA::~TestA(void)
{}

/*!
      \param fA an integer argument.
     \param fB an double argument.
     \return The test results
*/
double TestA::func(int fA, double fB)
{
      return fA*fB;

}
```

Snapshots from html type documentation pages after generating by Doxygen





A constructor.

~TestA (void)
A constructor.

double **func** (int fA, double fB)
A sample function.

Detailed Description

A test class.

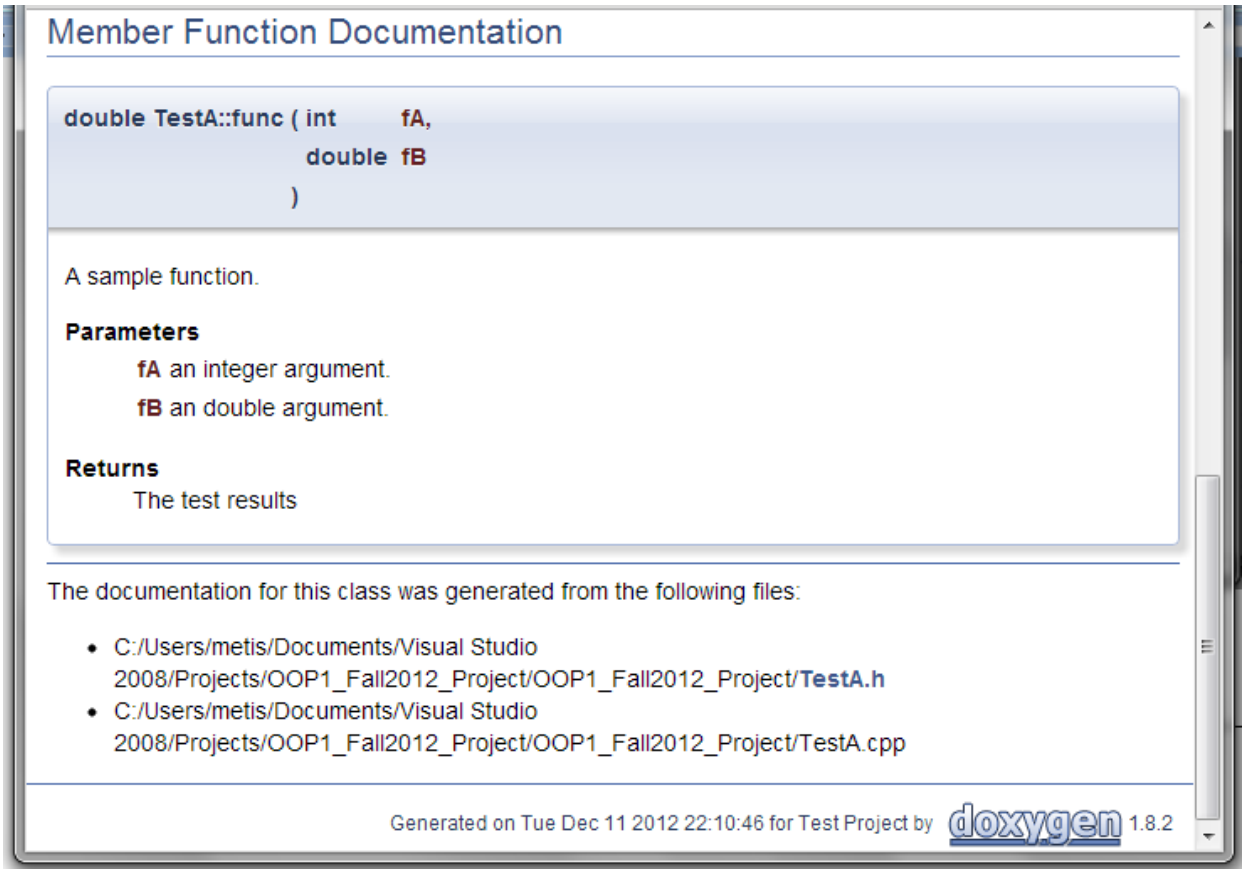
A more elaborate class description.

Member Function Documentation

double TestA::func (int fA,
double fB
)

A sample function.

Parameters



Member Function Documentation

double TestA::func (int fA,
double fB
)

A sample function.

Parameters

fA an integer argument.

fB an double argument.

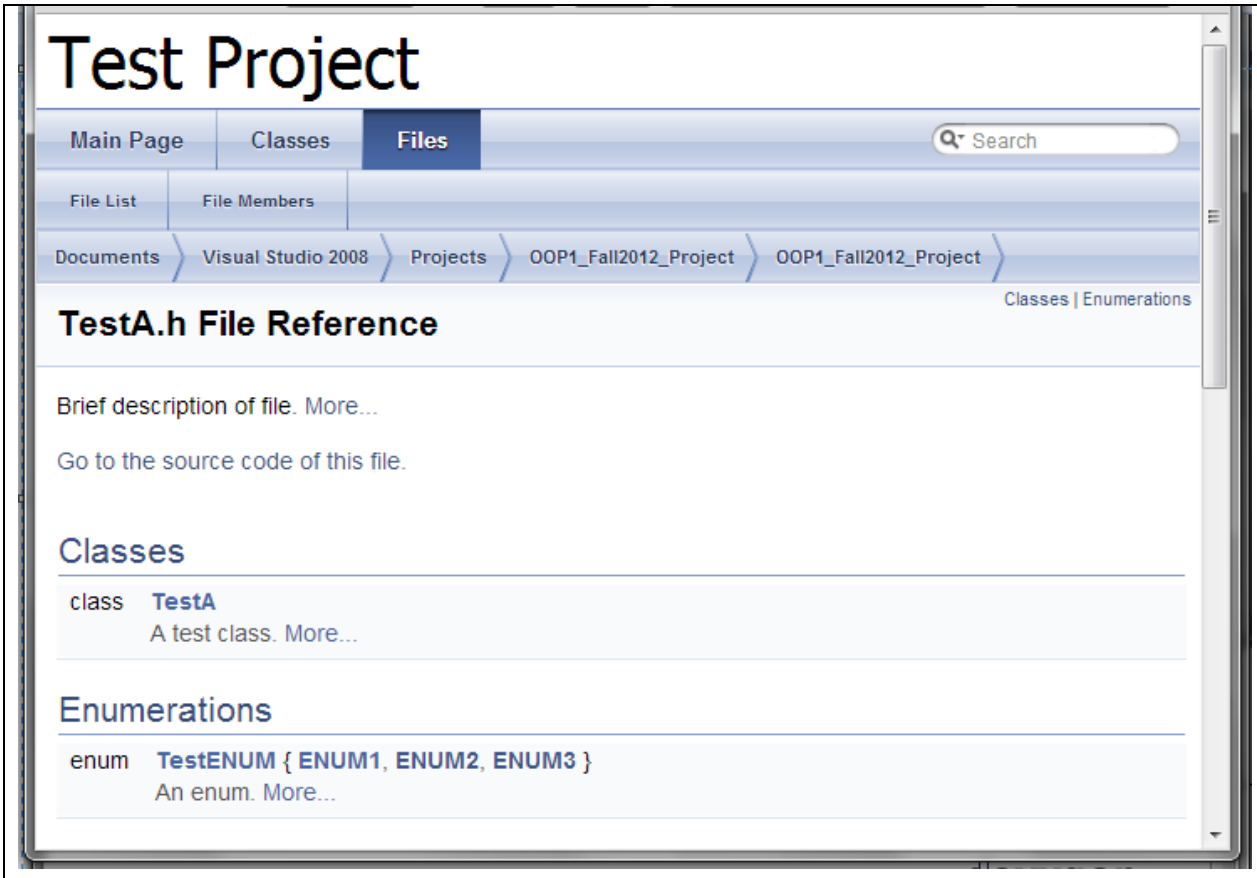
Returns

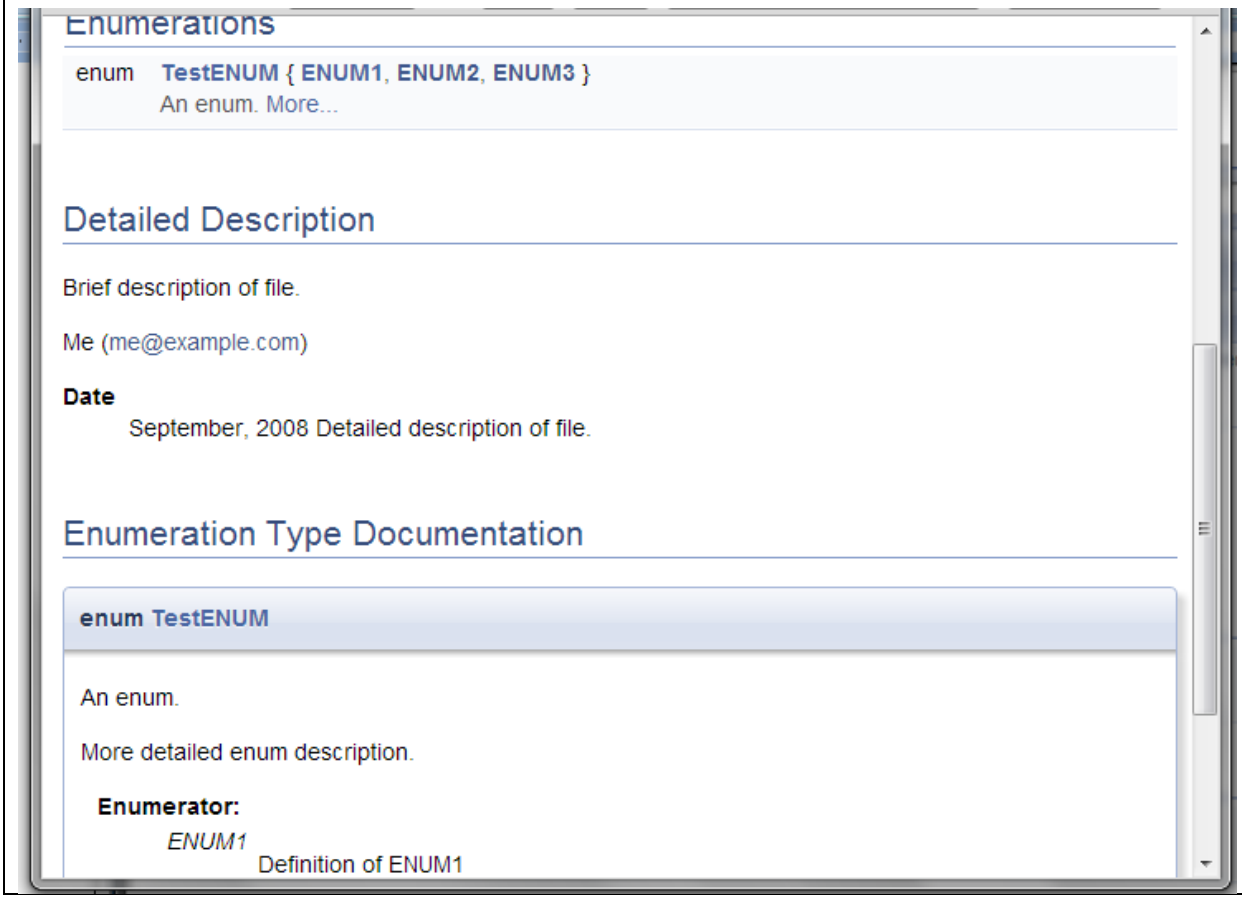
The test results

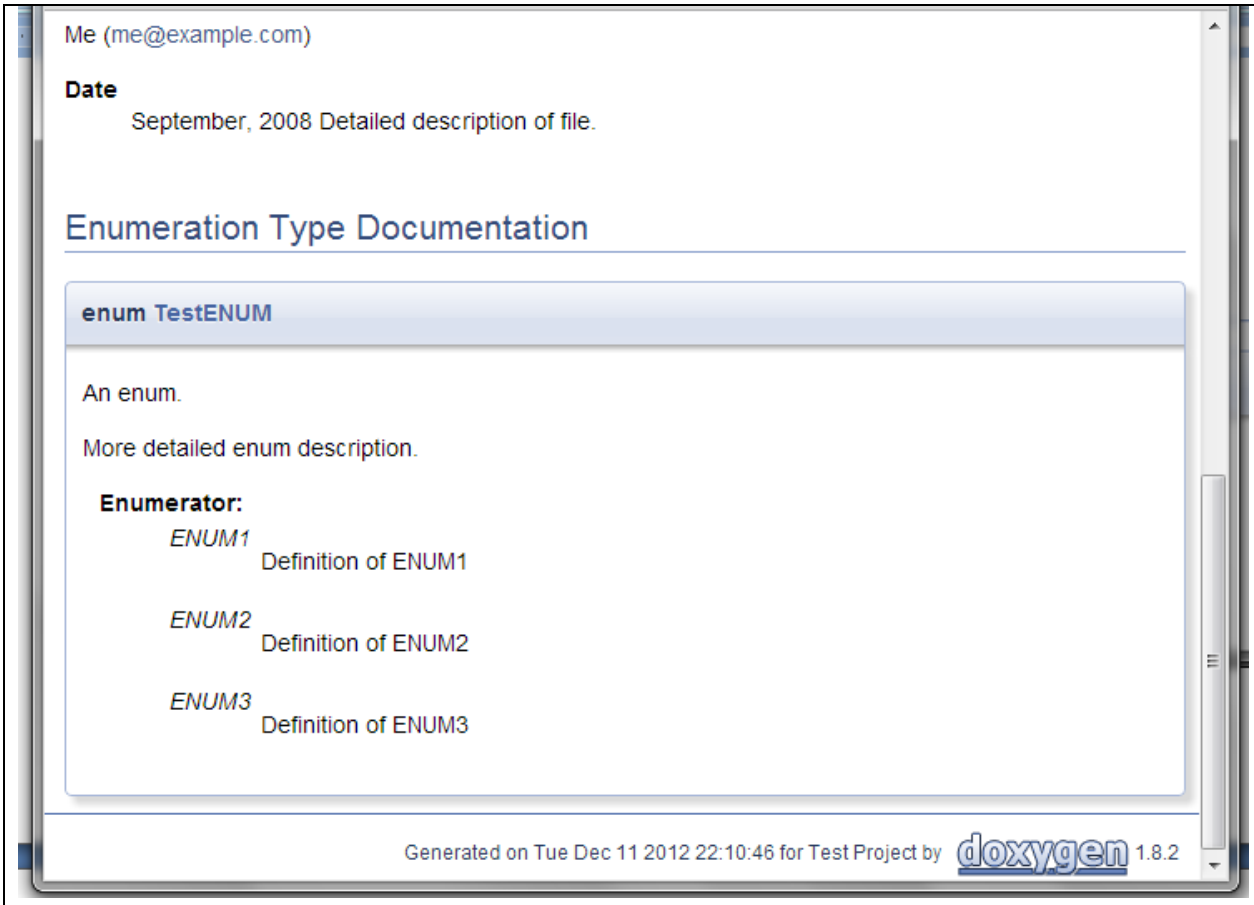
The documentation for this class was generated from the following files:

- C:/Users/metis/Documents/Visual Studio 2008/Projects/OOP1_Fall2012_Project/OOP1_Fall2012_Project/**TestA.h**
- C:/Users/metis/Documents/Visual Studio 2008/Projects/OOP1_Fall2012_Project/OOP1_Fall2012_Project/TestA.cpp

Generated on Tue Dec 11 2012 22:10:46 for Test Project by doxygen 1.8.2


Test Project
Main Page
Classes
Files
Search
File List
File Members
Documents
Visual Studio 2008
Projects
OOP1_Fall2012_Project
OOP1_Fall2012_Project
Classes | Enumerations
TestA.h File Reference
Brief description of file. More...
Go to the source code of this file.
Classes
class TestA
A test class. More...
Enumerations
enum TestENUM { ENUM1, ENUM2, ENUM3 }
An enum. More...


Enumerations
enum TestENUM { ENUM1, ENUM2, ENUM3 }
An enum. More...
Detailed Description
Brief description of file.
Me (me@example.com)
Date
September, 2008 Detailed description of file.
Enumeration Type Documentation
enum TestENUM
An enum.
More detailed enum description.
Enumerator:
ENUM1
Definition of ENUM1


Me (me@example.com)
Date
September, 2008 Detailed description of file.
Enumeration Type Documentation
enum TestENUM
An enum.
More detailed enum description.
Enumerator:
ENUM1
Definition of ENUM1
ENUM2
Definition of ENUM2
ENUM3
Definition of ENUM3
Generated on Tue Dec 11 2012 22:10:46 for Test Project by doxygen 1.8.2

A.2 Rapor Formatı

Kapak Sayfası

ESKISEHIR OSMANGAZI UNIVERSITESI
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

NESNE TABANLI PROGRAMLAMA I
PROJE RAPORU

Proje Başlığı

Proje grup üyesinin Nosu ve Adı-Soyadı
Proje grup üyesinin Nosu ve Adı-Soyadı
Proje grup üyesinin Nosu ve Adı-Soyadı
...

Ocak 2021

Rapor İçeriği

1. **Giriş**
   Proje ile ilgili genel bilgi

2. **Tasarım**
   Alt başlıklar eklenebilir. Tasarımla ilgili açıklamalar eklenecektir. UML diyagramları, grup üyelerine görev atamaları (kim hangi sınıfları kodladı), örnek girdiler için örnek program çıktıları, vb.

Görev Planlama (Mutlaka olmalı)

Takım Üyesi	**Görevleri**
Adı-Soyadı	*Class(es),menus, documentation, etc. . . .*
Adı-Soyadı	*. . .*
. . .	**. . .**

3. **Sonuçlar ve Değerlendirme**
   Proje sonuçlarının değerlendirilmesi, takım çalışması ile ilgili yorumlar, artılar ve eksiler, öneriler, vb.

A.3 Check list for the materials which should be submitted and grading

CHECK LIST

Materials	**Included**
Header and source files of each classes and application (.h, .cpp)	Yes / No
Each class has separate header and source files?	Yes / No
Codes are well formatted?	Yes / No
Codes are commented by using Doxygen tags?	Yes / No
There are test programs for each class?	Yes / No
Internal documentation generated by Doxygen (.htm)	Yes / No
Report	Yes / No

GRADING (TENTATIVE)

Items	**Grade (%)**
Individual Studies	
Internal Documentation	10
Code quality (well formatted)	10
Completeness	10
Overall (including all classes and application)	
Internal documentation (using doxygen)	15
Overall code quality	10
Group work	15
Functionality (implemented and correctly running functionalities).	10
Report	20