152114002: NESNE TABANLI PROGRAMLAMA I
2017-2018 GÜZ DÖNEMİ
DÖNEM PROJESİ
Son Teslim Tarihi: 07 Ocak 2018, Pazar, 17:00

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, 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.**

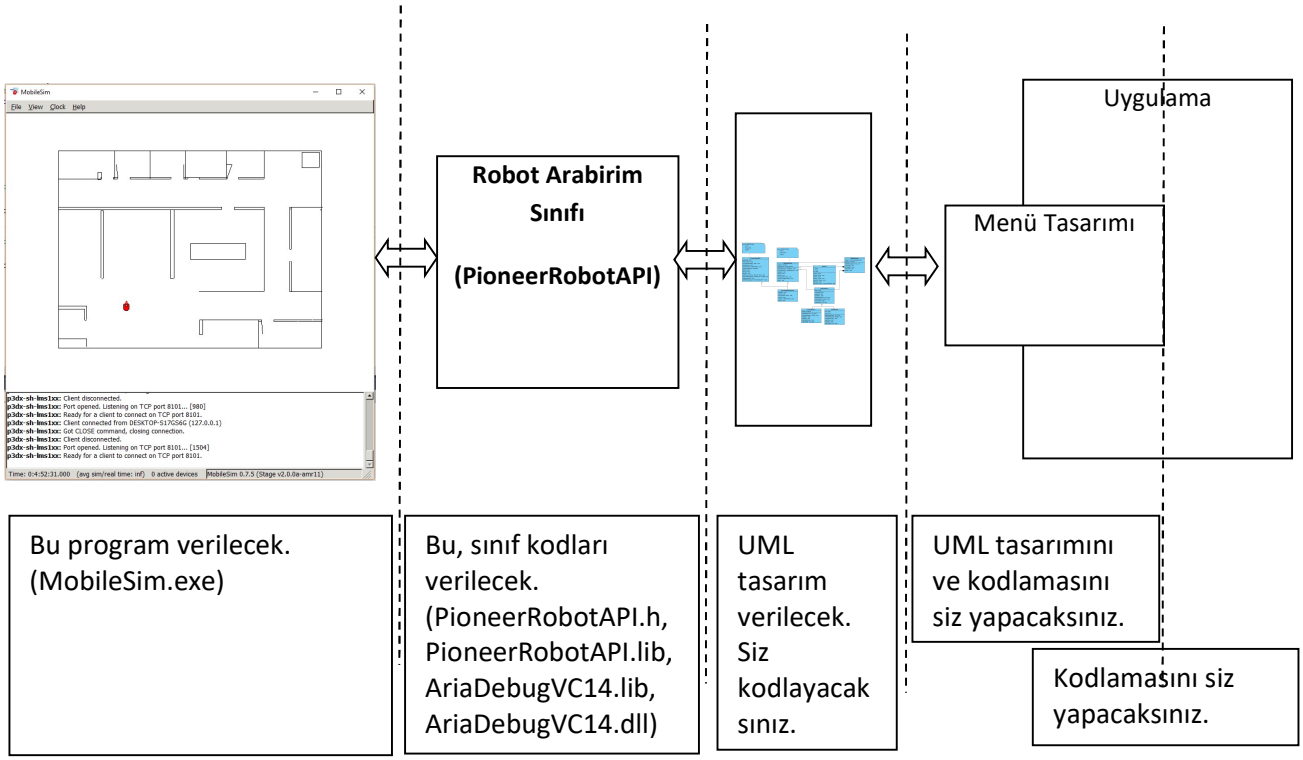
10. **Dönem sonunda, proje gruplarının sunum yapması gerekmektedir.**

PROJE BAŞLIĞI: Gezgin Robot Denetim Sistemi

Proje Açıklaması:

Proje konusu, 2B gezgin robot simülatöründeki robotun denetimi (hareket ettirilmesi ve sensörlerinden veri alınması) için yazılım geliştirmektedir. Bu projede, size bir simulator programı ve bu programdaki robotu denetleyebilmek için gereken sınıf kütüphanesi verilmektedir.

Şekil 1' de görüldüğü gibi, Robot simülatörü size verilecektir. Ayrıca, http://robots.mobilerobots.com/MobileSim/download/current/MobileSim-0.7.5.exe linki üzerinden de yükleyebilirsiniz. Bu program, açılırken robotun hareket edeceği ortamı sağlayan bir harita dosyasını yüklemeniz gerekmektedir. Size verilen "office.map" dosyasını yükleyebilirsiniz. Program arayüzünde mobil robot kırmızı renkte görünecektir. Bu kırmızı renkteki robot, yazılan programlarla hareket ettirilebilmektedir. Programa erişim, TCP/IP protokolü ile haberleşerek yapılmaktadır. Siz bu karmaşık programlama yapıları ile uğraşmayacaksınız. Bu karmaşık kodlamalar, PioneerRobotAPI adlı sınıfın altında gerçekleştirilmektedir. Siz uygulama geliştirirken simülatördeki robota erişim için bu sınıfı kullanacaksınız. Bu sınıfı projenizde kullanabilmeniz ve projenizi derleyebilmeniz için (PioneerRobotAPI.h, PioneerRobotAPI.lib, AriaDebugVC14.lib, AriaDebugVC14.dll, Aria dizini) dosyaları size verilmektedir. Ayrıca, PioneerRobotAPI sınıfının üye fonksiyonlarının açıklamaları PioneerRobotAPI.h dosyasında vardır. Ayrıca, size verilen RobotAPITest.cpp programı da sınıfın kullanımında size yardımcı olacaktır.



Şekil 1.

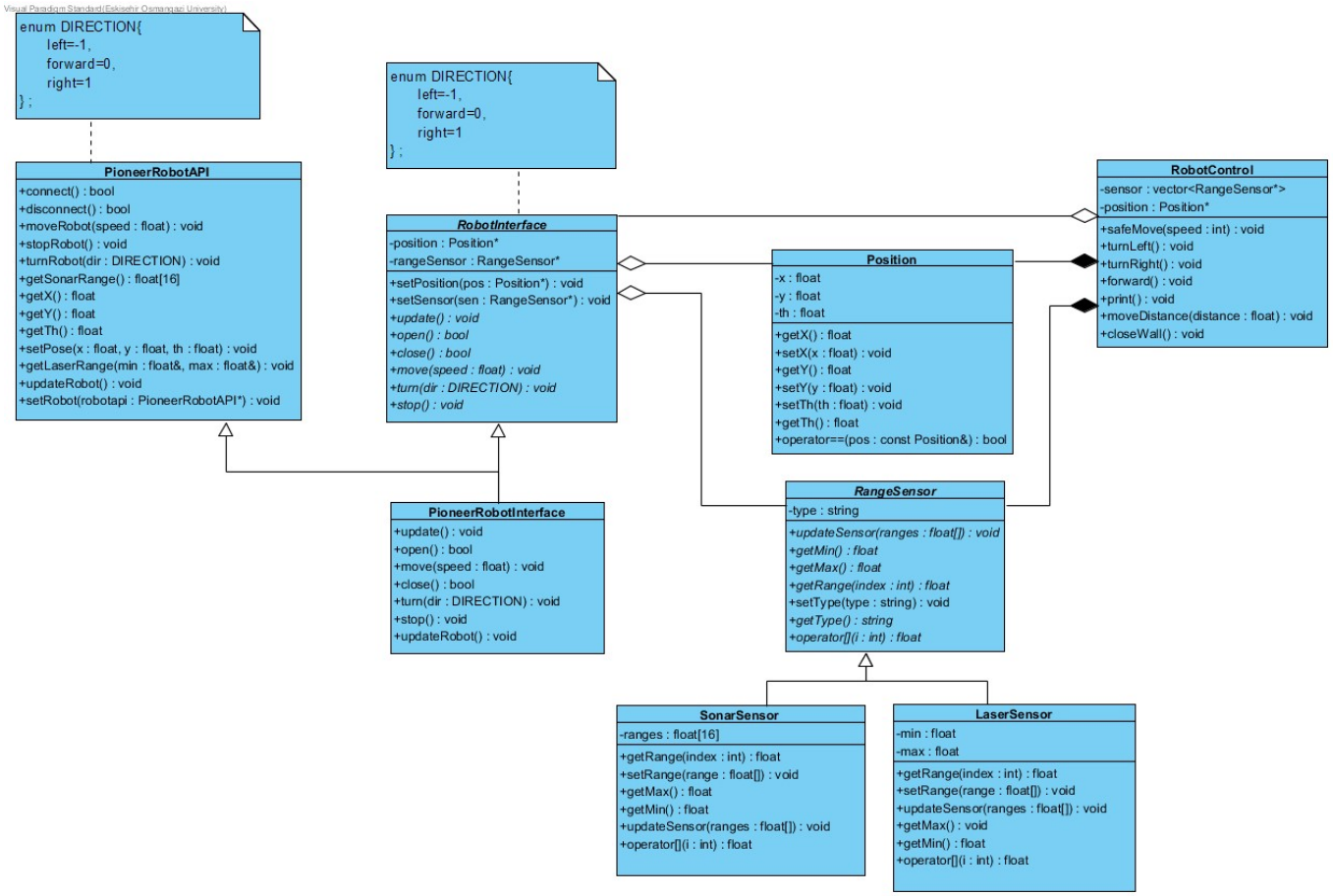
Önerimiz ilk önce şu adımları gerçekleştirmenizdir.

1. MobileSim programını kurun ve çalıştırın.
2. MobileSim programını çalıştırırken, office.map dosyasını yükleyin.
3. AriaDebugVC14.dll dosyasını, c:\Windows\SysWOW64 dizini altına kopyalayın.

4. Visual Studio 2015 (farklı sürüm kullanıyorsanız, o sürüme uygun AriaDebugVC14.lib, AriaDebugVC14.dll dosyalarını isteyiniz) altında bir proje C++ Empty Project yaratınız.
5. Projeye, RobotAPITest.cpp dosyasını ekleyiniz.
6. Proje ayarlarını yapınız.
    a. Property Pages altında, C/C++ altında, General altında, Additional Include Directories 'e PioneerRobotAPI.h dosyasının ve Aria dizininin yollarını ekleyiniz.
    b. Property Pages altında, Linker altında, General altında, Additional Library Directories 'e PioneerRobotAPI.lib ve AriaDebugVC14.lib dosyalarının yollarını ekleyiniz.
    c. Property Pages altında, Linker altında, Input altında, Additional Dependencies 'e PioneerRobotAPI.lib ve AriaDebugVC14.lib dosya adlarını ekleyiniz.
7. Projeyi derleyiniz ve çalıştırınız.
8. Eğer, program çalışırken MobileSim programındaki robot hareket ediyorsa ve ekranda robotla ilgili bilgiler görünüyorsa, başarılı oldu anlamına gelmektedir.
9. Bundan sonra, projede istenen diğer kaynak dosyaları ekleyip kodlamalarınızı yapabilirsiniz.

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 PioneerRobotAPI: Bu sınıf size simülatör ile birlikte 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, PioneerRobotAPI.h dosyasında bulunabilir.

Class RobotInterface: Bu sınıf, soyut (abstract) sınıftır. Tasarımların, simülatöre erişimi için arayüz görevi görmektedir. Şu anda Pioneer simülatörüne arayüz sağlamaktadır. Ancak, ilerde farklı robot simülatörleri için PioneerRobotInterface sınıfı gibi sınıflar eklenerek onlar içinde arayüz sağlayacaktır.

update(): Saf soyut sınıftır. Konum ve sensör bilgilerinin güncellenmesinde kullanılmaktadır.

open(): Saf soyut sınıftır. Robot simülatörü ile bağlantıyı açar ve robota erişebilir duruma getirir.

close(): Saf soyut sınıftır. Robot simülatörü ile bağlantıyı kapatır

move(speed): Saf soyut sınıftır. Parametre olarak verilen hızda (mm/s), robotun ilerlemesi sağlanır. "-" değerlerde geriye doğru gider.

turn(dir): Saf soyut sınıftır. Parametre olarak verilen yönde robotu döndürür.

stop(): Saf soyut sınıftır. Robotu durdurur.

Class PioneerRobotInterface: Bu sınıf, Pioneer robot simülatörünün fonksiyonlarını kullanarak, bizim tasarımımıza arayüz sağlamaktadır. RobotInterface sınıfındaki, saf soyut fonksiyonları, PioneerRobotAPI sınıfındaki fonksiyonlardan faydalanarak gerçekler.

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

Class RangeSensor: Soyut (Abstract) sınıftır. Mesafe ölçen farklı tip sensörler için arayüz oluşturur.

updateSensor(ranges): Sensör mesafe değerlerini, parametre ranges ile verilen değerle ile gündeller.

getMin(): Mesafe değerlerinden minimum olanı döndürür.

getMax(): Mesafe değerlerinden maksimum olanı döndürür.

getRange(i): i. Sensörün mesafe bilgisini döndürür.

setType(type): sensörün tipini atar.

getType(): Sensörün tipini döndürür.

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

Class SonarSensor: Mesafe ölçen farklı tip sensörler için arayüz oluşturur.

updateSensor(ranges): Sensör mesafe değerlerini, parametre ranges ile verilen değerle ile gündeller.

getMin(): Mesafe değerlerinden minimum olanı döndürür.

getMax(): Mesafe değerlerinden maksimum olanı döndürür.

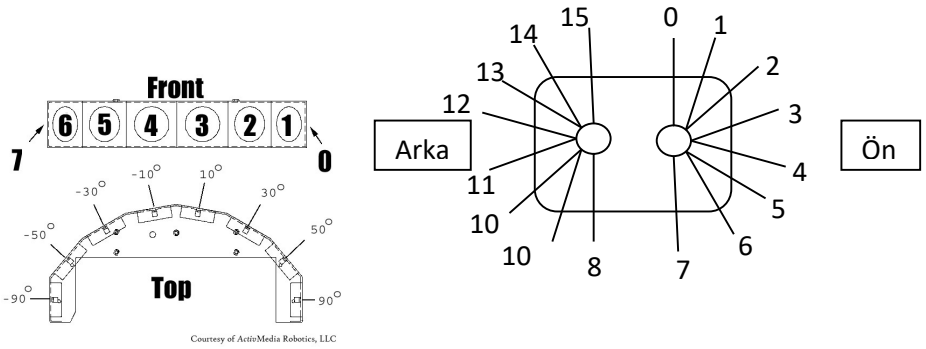
getRange(i): i. Sensörün mesafe bilgisini döndürür.

setType(type): sensörün tipini atar.

getType(): Sensörün tipini döndürür.

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

Şekil 3'de sonar sensörlerin robot üzerindeki yerleşimi ve indeksleri verilmektedir.



Şekil 3.

Class RobotControl: Robot denetiminde kullanılan bir sınıftır.

safeMove(speed): Bu fonksiyon, istenen hızda robotun ilerlemesini sağlar. Ancak, robotun önünde engel varda ilerlemesini durdurur.

turnLeft(): sola döndürür.
turnRight(): sağa döndürür.
forward(): düz gitmesini sağlar.
print(): robotun konumunu ve sensor değerlerini ekrana belli bir formatta yazdırır.
moveDistance(distance): robotun verilen mesafe kadar gitmesini, mesafeyi katettikten sonra durmasını sağlar.
closeWall(): Robot düz gider ve bir duvara belli bir uzaklığa geldiğinde durur.

Kısım 2.

Kısım 1'deki sınıfları kullanacak bir menu tasarlayınız. Menuler, metin tabanlı olacaktır. Grafik arayüz ve menüler istenmemektedir. 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 3.

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

DOSYA EKLERİ:

EK 1. Simulator Programı ve Office.map dosyası

EK 2. PioneerRobotAPI.h, PioneerRobotAPI.lib, AriaDebugVC14.lib, AriaDebugVC14.dll, Aria dizini.

EK 3. RobotAPITest.cpp test program kodu.

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:

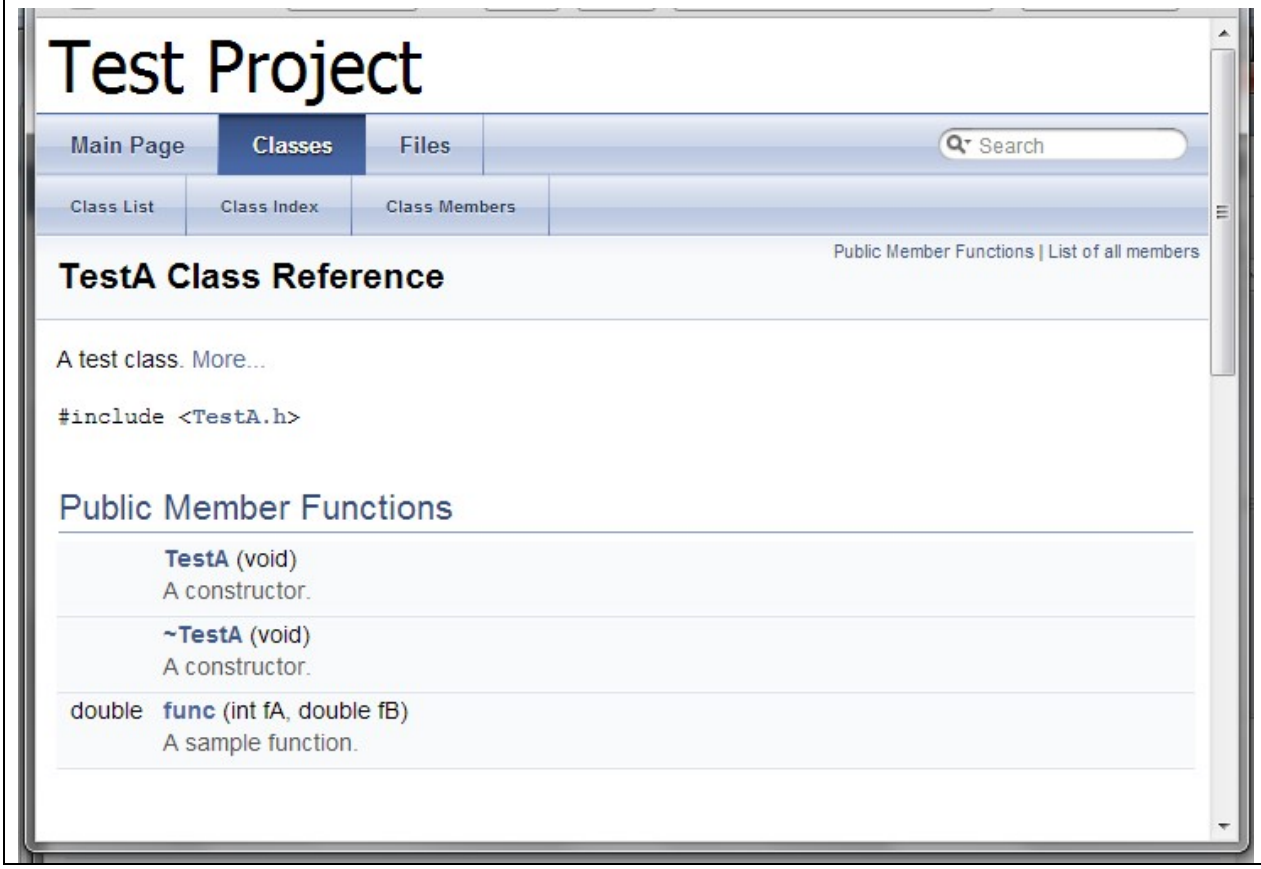
```
/**
 * @file   TestA.h
 * @Author Me (me@example.com)
 * @date   September, 2008
 * @brief  Brief description of file.
 *
 * Detailed description of file.
 */

//! 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.
```

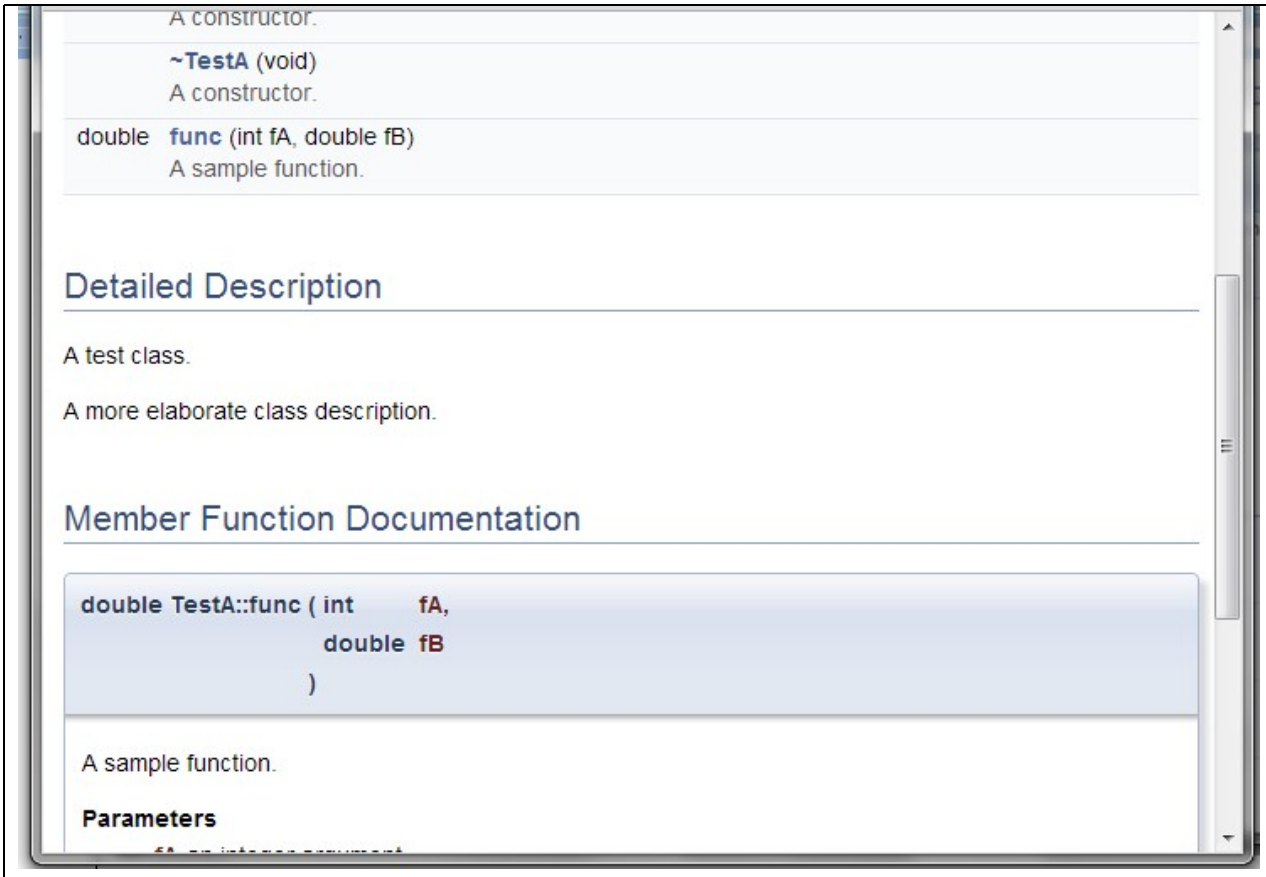
```
    ~TestA(void);
    //! A sample function.
    double func(int fA, double fB);
};
```

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.

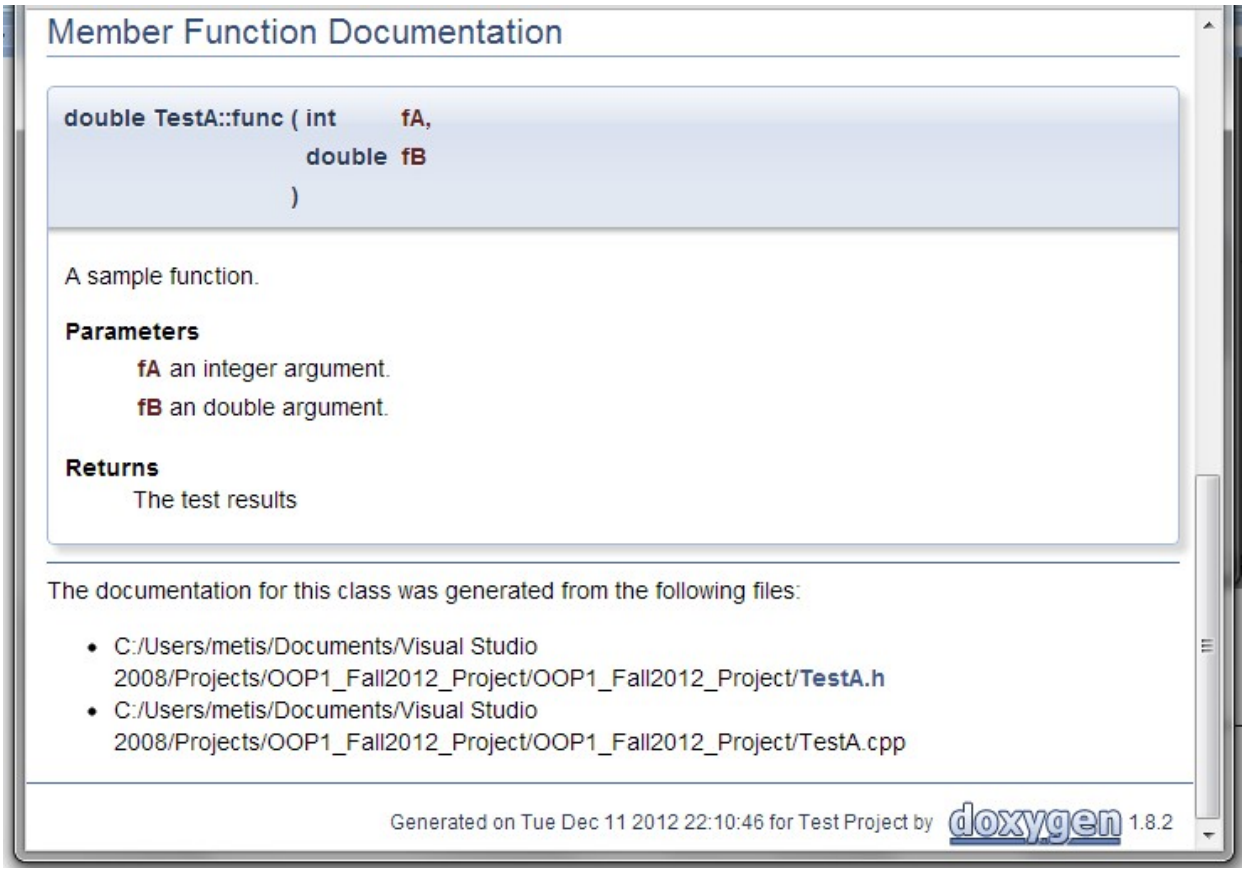
A more elaborate class description.

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.

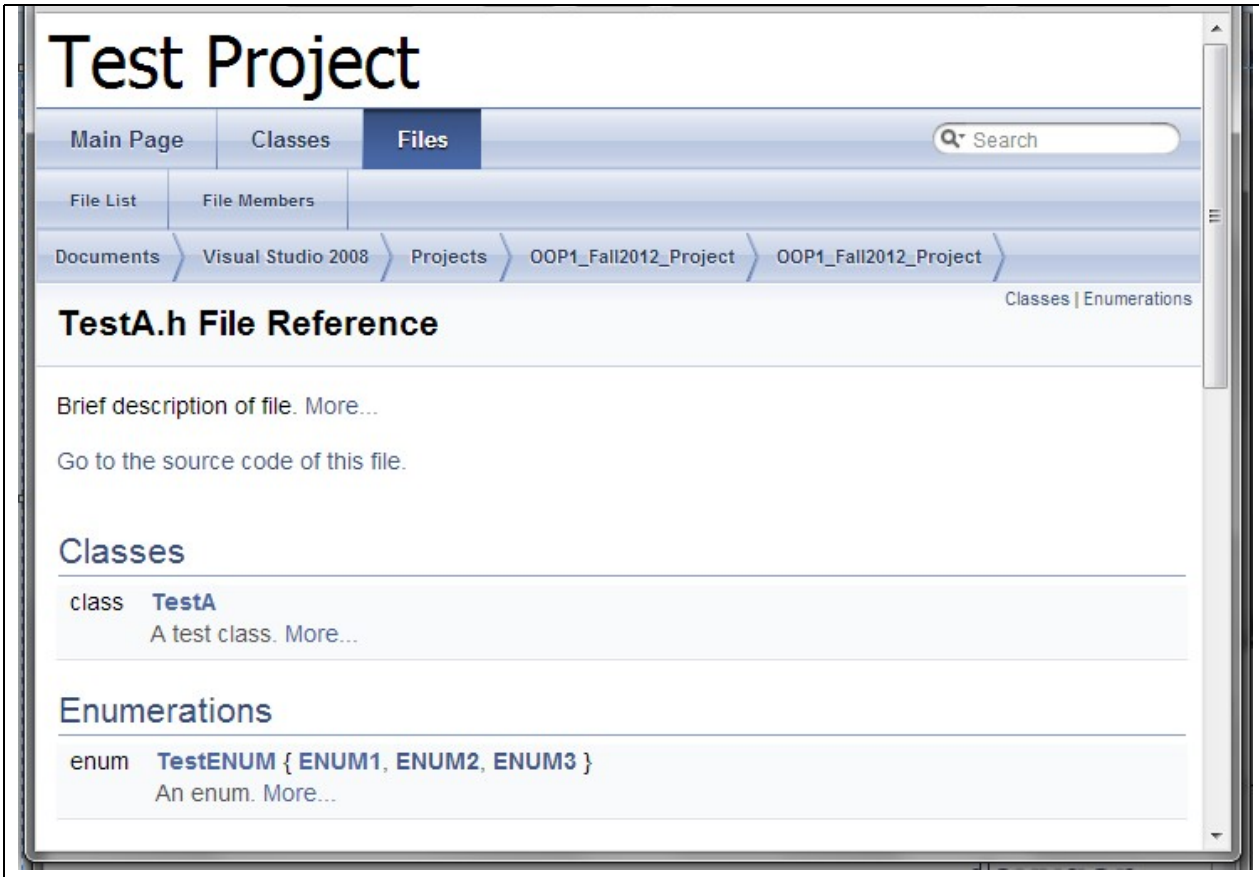
Returns

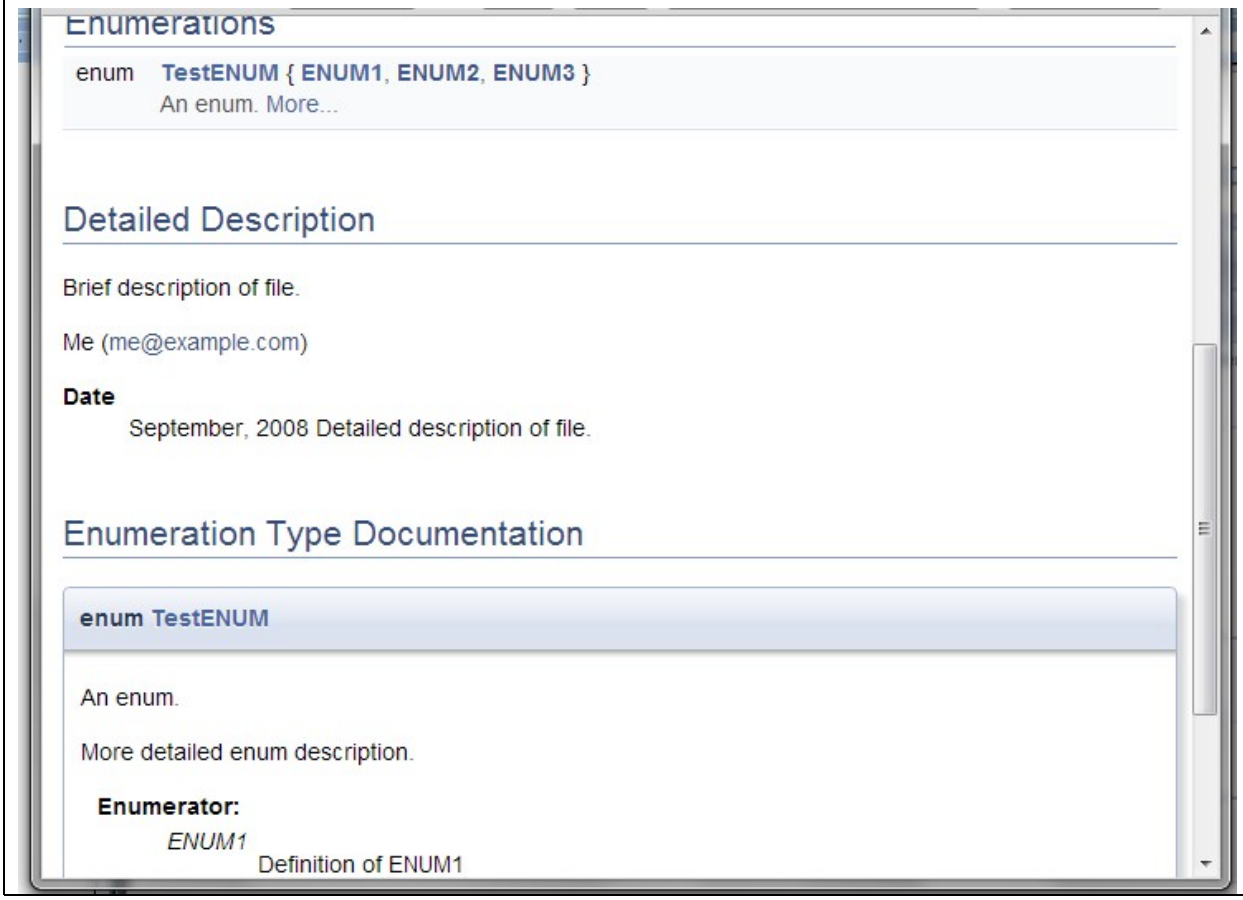
The test results

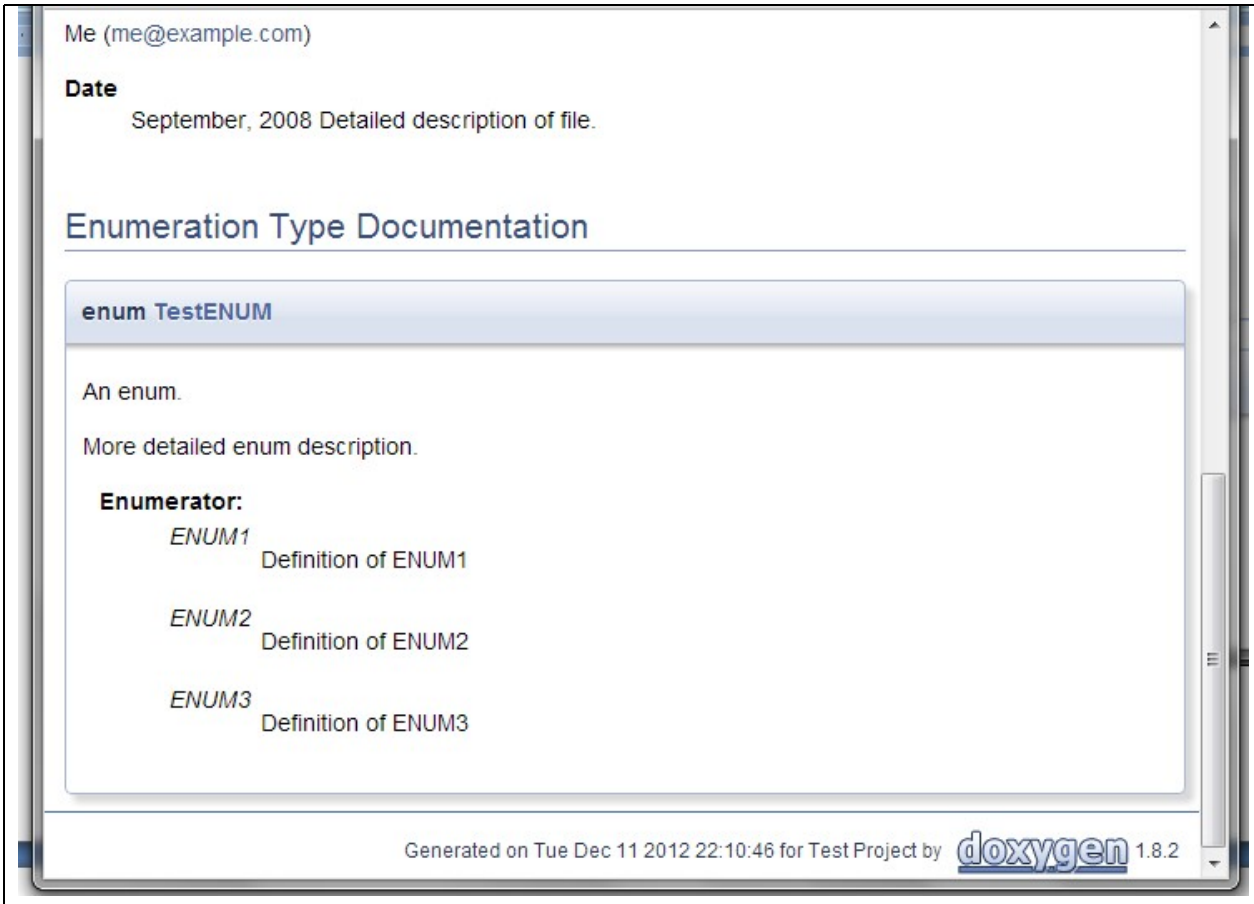
The documentation for this class was generated from the following files:

- 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 Report Format

Cover Page

ESKISEHIR OSMANGAZI UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

OBJECT ORIENTED PROGRAMMING I
DESIGN PROJECT REPORT

Project Title

Project Group Member ID and Name
Project Group Member ID and Name
Project Group Member ID and Name
...

January 2015

Report Contents

1. **Introduction**
   General information about project

2. **Design**
   You may add subtitles. This part includes explanations about design, UML diagrams, task assignments, sample outputs of the application program for the sample inputs, etc.

Task Assignments (Must be included)

Group Member	**Tasks**
Name of Group member	*Class(es),menus, documentation, etc. . . .*
Name of Group member	*. . .*
. . .	**. . .**

3. **Conclusion**
   Evaluation of the project results, comments about the team work, pros and cons, advices for further works, etc.

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