152113017: NESNE TABANLI PROGRAMLAMA I
2024-2025 GÜZ DÖNEMİ
DÖNEM PROJESİ (II. AŞAMA)
Son Teslim Tarihi: 29 Aralık 2024, Pazar, 23:00

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, 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/veya kodlarınızın kısmen ya da tamamen başka gruplardan ve referans gösterilmemiş kaynaklardan alındığı belirlenirse, sıfır not alırsınız.**

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

PROJE BAŞLIĞI: Her Yöne Hareketli (Omni-directional) Robot Denetim Sistemi (2nci Aşama)

Proje Açıklaması:

İlk aşamada yazılan sınıflar, Şekil 1'de verilen UML Sınıf diyagramına göre revize edilecektir. Yapılacak değişiklikler, aşağıda açıklanmakta olup, verilen tasarımda eksik olan ya da çelişen durumlar sizler tarafından giderilecektir.

- Gri renkte verilen bloklar için bir şey yapmayacaksınız. Bu kısımlar size, sadece yazılımın farklı bir robot platformu için geliştirildiğinde ekleme yapılacak kısımları göstermektedir. Yeni tasarımda görüldüğü gibi, yeni robotların da yazılıma dahil edilmesinde soyutlama yapılmaktadır.

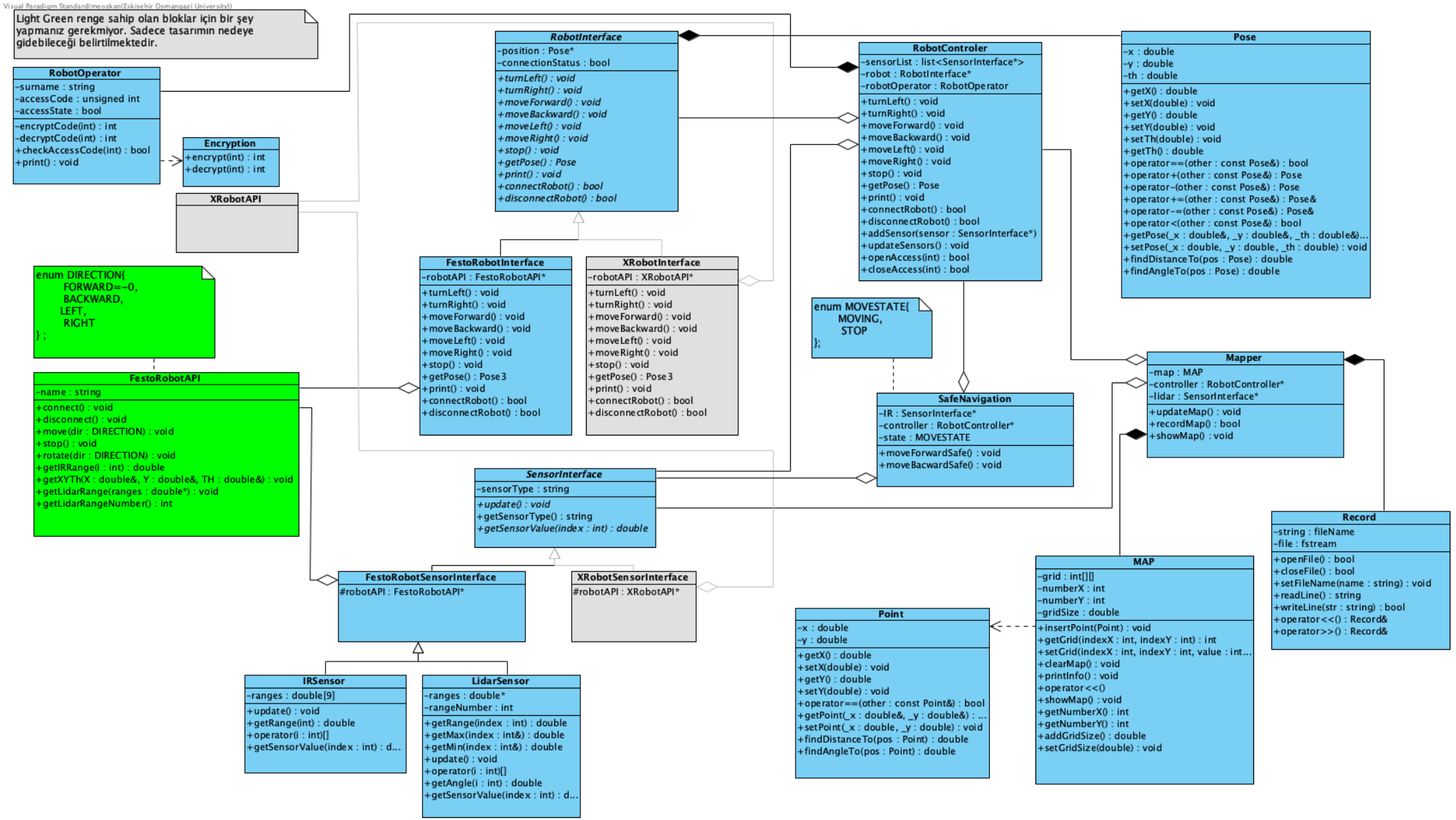
- `RobotControler` ve `FestoRobotAPI` arasına `RobotInterface` adında bir sınıf eklenmiştir. `RobotInterface` abstract bir sınıf olup, `FestoRobotAPI` ile doğrudan ilişkili olan `FestoRobotInterface` için bir arayüz oluşturmaktadır. Böylece farklı bir robota ait bir API ya da aynı robota ait farklı bir API kullanıldığında, `RobotInterface` sınıfından miras alan başka bir sınıf eklenecektir. Bu eklentiden, yazılımımızın geri kalanı etkilenmeyecektir.

- `RobotInterface` abstract sınıfına `sensorList` üyesi eklenmiştir. `addSensor` üye fonksiyonu ile sensörlerin adresleri bu listeye eklenmektedir. `updateSensors()` üye fonksiyonu çağrıldığında, listeye eklenen tüm sensör nesnelerinin `updateSensor` fonksiyonları çağrılmak suretiyle tüm sensör verilerinin tek bir yerden güncellenmesi mümkün olabilmektedir.

- `IRSensor` ve `LidarSensor` sınıfları, `FestoRobotSensorInterface` sınıfı altında toplanmıştır. `FestoRobotSensorInterface` sınıfı da `SensorInterface` abstract sınıfı altındadır. Yeni bir robot için tanımlı sensörler olduğunda onlarda `SensorInterface` altında eklenecektir. Sensör sınıfları arasına iki yeni sınıf eklenmiştir. Bunlar `getSensorType()` ve `getSensorValue()` sınıflarıdır. `getSensorType()` fonksiyonu sensör tipini `string` olarak döndürmektedir (Örneğin, "IR", "lidar" etc.). `getSensorValue(i:int)` fonksiyonu ait olduğu sensörün i indeksindeki verisini `double` olarak döndürmektedir.

- `RobotControler` sınıfı robotla ilgili işlemleri `RobotInterface` ve `SensorInterface` üzerinden işletecektir. Bunun yanında yeni fonksiyonlarda eklenmiştir. Bunlar;
  - `addSensor()` fonksiyonu, listeye sensör adreslerini eklemek için kullanılır.
  - `updateSensor()` fonksiyonu, listedeki tüm sensör nesnelerinin update fonksiyonu çağrılmak suretiyle, tek noktadan tüm sensör verilerinin güncellenmesi sağlanır.
  - `openAccess()` fonksiyonu, erişim için kullanılmaktadır. Bu fonksiyon üzerinden, doğru şifre giriş yapılmadığı takdirde, `RobotControl` sınıfının hiçbir fonksiyonu bir işlem yapmayacaktır. Tüm fonksiyonlar çağrıldığında işlem yapmadan dönecektir. Bu fonksiyon kullanılarak doğru şifre verildi ise, tüm üye fonksiyonlar yapması gereken işlemi yapacaktır. (`RobotControl` sınıfı altına bir `boolean` değişken tanımlayıp, şifrenin girilip/girilmediğine göre `true/false` yapılıp, tüm fonksiyonların bu değişkeni kontrol etmesi sağlanabilir.)
  - `closeAccess()` fonksiyonu ile doğru şifre verildiğinde, erişim tekrar kapatılacaktır. Tekrar açılana kadar fonksiyonlar işlemlerini yapmayacaktır.

- `Point`, `MAP`, `Record`, `Pose`, `RobotOperator` , `Encription` sınıflarında bir değişiklik olmamakla birlikte, `SafeNavigation` ve `Mapper` sınıflarının Sensör ilişkileri `SensorInterface` üzerinden sağlanmaktadır.



Şekil 1. Projenin Revize Tasarımı (UML Sınıf Diyagramı)

I. Daha önce yazılan sınıflar, bu tasarıma göre revize edilecektir.

II. İlk aşamada yazılan menü uygulamasının benzeri olarak, revize edilen RobotControler sınıfında bulunan tüm fonksiyonları kullanabilecek şekilde yeni bir tane yazılacaktır.

III. Yazılan menü uygulaması çalıştırılacak, örnek bir kullanım sonrasında ekrandaki tüm hareketler rapora eklenecektir.