**ROBOT PROJESİ**

*Hacı Bayram ÜNAL*

*180201006*

Bilgisayar Mühendisliği Bölümü Kocaeli Üniversitesi

hacibayram.unal@hotmail.com

# Özet

Bu projenin amacı belirli robot türleri için ayrı ayrı sınıf oluşturmak yerine sistem yönetilebilirliğini sadeleştiren bir uygulama geliştirmektir. Bütün robotların ortak özellikleri bulunmaktadır. Bu özellikleri tekrardan her sınıf için tanımlamak yerine, nesneye yönelik programlama mantığı ile bu robot sınıfından türeyen sınıflara da benzer özellikler kalıtılır ve projenin yönetilebilirliğini kolaylaştırılır.

Robotların ortak olan özelliklerinin kalıtılması, soyutlama(abstraction)[1], kalıtım(inheritance)[2], kapsülleme (encapsulation)[3], çok biçimlilik (polymorphism)[4] yapıları ile sağlanmaktadır. Kalıtılan özelliklere ek olarak, eğer oluşturulacak robotun kendi özelliği varsa, bu özellik o sınıfa ve bu sınıftan kalıtım alınan diğer sınıflara da ait olmaktadır.

Proje Java programlama dili kullanılarak geliştirilmiştir. Robot hareketleri, engellerin ve robotun son konumunun kullanıcıya gösterilmesi için ‘SWING’[5] kütüphanesi kullanılmıştır.

**Anahtar kelimeler :** inheritance, abstraction, encapsulation, polymorphism, java, swing

# Giriş

Bu projede, belirli robot çeşitleri ve bu robotların nitelikleri de dikkate alınarak kullanıcı isteklerine göre davranışlarını gerçekleştiren bir program hedeflenmiştir.

Proje geliştirilirken JAVA programlama dili, kullanıcı girdilerine göre robotun hareketlerini belirledikten sonra grafik ekranında göstermek üzere ‘SWING’ kütüphanesi kullanılmıştır.

Proje belirlenen robot türlerinin, kullanıcının isteklerine göre nitelikleri belirlendikten sonra birbirleri arasında ve hareket edeceği yol üzerinde hareket ederken nasıl davranması gerektiğini işlemektedir.

# Yöntem

Belirtilen robotların, nesneye yönelik programlama mantığı ile birbirleri arasında ilişki kurulması sağlandı. Her robot sınıfında ortak olan özellikler kalıtım ile alt sınıflara aktarıldı. Sınıfların ek olarak ihtiyaç duyduğu nitelikler tanımlandı. Robotlar türlerine ayrıldığı için robotların türlerini belirtmek üzere arayüz(interface) yapısı kullanıldı. Bu yapı, kullanıldığı robotun türünü belirtir ve kullanması zorunlu kılınan metotları oluşturmasını ister.

Program başlatıldığında kullanıcı öncelikle oluşturacağı robot sayısını girmelidir. Ardından sırasıyla oluşturacağı robotların türlerini ve niteliklerini girmelidir. Girilen robotlar ve niteliklerine göre bazı koşullar oluşturulmuştur (bazı robot türleri diğerlerinden daha ağır yük kaldıramaz veya diğerlerinden daha hızlı olamaz gibi).

Robot türleri arasında hız açısından artan sırada spider robot, paletli robot, tekerlekli robot ilişkisi vardır. Yani en hızlı tekerlekli robot, en yavaş spider robottur. Bu hiyerarşinin bozulmaması için, yani normalde yavaş olan robotun hızlı olan robottan daha yüksek değer almaması için kontrol yapıldı.

**Sözde kod :**

i = 0; i < robotlar.length olduğu sürece

Eğer robotlar[i] instanceof GezginRobotlar ise

Eğer robotlar[i] instanceof SpiderRobot ise

j = 0; j < robotlar.length olduğu sürece

Eğer robotlar[j] instanceof GezginRobotlar ise

Eğer robotlar[j] instanceof TekerlekliRobot ise

int tekerlekliHız = (TekerlekliRobot)robotlar[j].gezinmeHızı;

int spiderHız = (SpiderRobot)robotlar[i].gezinmeHizi;

Eğer spiderHız > tekerlekliHız ise

spiderHiz > tekerlekliHiz olduğu sürece

Eğer spiderHız <= tekerlekliHız ise

break;

Değilse

Println(“Tekerlekli robot spider robottan yavaş olamaz”);

Println(“1: tekerlekli robotun hızını değiştir”);

Println(“2: spider robotun hızını değiştir”);

int seçim = okuyucu.nextInt();

Eğer seçim = 1 ise

Println(“tekerlekli robotun yeni hızını girin:”);

tekerlekliHız = okuyucu.nextInt();

(TekerlekliRobot)robotlar[j].gezinmeHızı = tekerlekliHız;

Degilse eğer seçim = 2 ise

Println(“spider robotun yeni hızını girin”);

spiderHız = okuyucu.nextInt();

(SpiderRobot)robotlar[i]. gezinmeHızı = spiderHız;

Değilse

Println(“Yanlış giriş yaptınız”);

Değilse

başa dön;

Manipülatör robotlar arasında da yük taşıma özelliği için bir ilişki bulunmaktadır. Bu ilişkiye göre seri robotlar paralel robotlardan daha ağır yükler kaldıramazlar. Robotlar oluşturulurken bu kontrol de yapılır ve ilgili robot için uygun değer alınmazsa, doğru değer alınana kadar kullanıcıya yük taşıma değeri tekrar sorulur.

**Sözde kod :**

i = 0; i < robotlar.length olduğu sürece

Eğer robotlar[i] instanceof GezmeyenRobotlar ise

Eğer robotlar[i] instanceof SeriRobot ise

j = 0; j < robotlar.length olduğu sürece

Eğer robotlar[j] instanceof GezmeyenRobotlar ise

Eğer robotlar[j] instanceof ParalelRobot ise

int paralelYük = (ParalelRobot)robotlar[j].yükKapasitesi;

int seriYük = (SeriRobot)robotlar[i].yükKapasitesi;

Eğer seriYük > paralelYük ise

seriYük > paralelYük olduğu sürece

Eğer seriYük <= paralelYük ise

break;

Değilse

Println(“Seri robot paralel robottan ağır yük kaldıramaz”);

Println(“1: Seri robotun yük kapasitesini değiştir”);

Println(“2: Paralel robotun yük kapasitesini değiştir”);

int seçim = okuyucu.nextInt();

Eğer seçim = 1 ise

Println(“Seri robotun yeni yük kapasitesini girin:”);

seriYük = okuyucu.nextInt();

(SeriRobot)robotlar[j].yükKapasitesi = seriYük;

Degilse eğer seçim = 2 ise

Println(“Paralel robotun yeni yük kapasitesini girin”);

paralelYük = okuyucu.nextInt();

(ParalelRobot)robotlar[i].yükKapasitesi = paralelYük;

Değilse

Println(“Yanlış giriş yaptınız”);

Değilse

başa dön;

Ardından oluşan robotun konumunu (x,y) biçiminde kullanıcıdan alınır ve ızgaralar üzerine yerleştirilir.

Robotlar alındıktan sonra, eğer oluşturulan robotlar içerisinde gezgin robotlardan bir nesne varsa engel oluşturulabilir demektir. Eğer hepsi manipülatör ise engel almaya gerek olmayacaktır. Bunun için bir kontrol yapıldı.

**Sözde kod :**

boolean engelAlinacakMi = false;

i = 0; i < robotSayisi olduğu sürece

Eğer robotlar[i] instanceof Gezgin ise

engelAlinacakMi = true;

break;

i’yi bir arttır başa dön

Eğer robotlar arasında gezgin robot varsa engel olabilir demektir. Eğer engel yerleştirilecekse engel sayısı ve engel sayısına göre koordinatları alınır, engel alınmayacaksa program akışına devam eder.

**Sözde kod :**

Eğer engelAlinacakMi = true ise

Println(“Engel yerleştirilecek mi? Evet:1 Hayır:0”);

int engelBilgisi = okuyucu.nextInt();

Eğer engelBilgisi = 1 ise

Println(“Kaç engel yerleştirilecek?”);

int engelSayisi = okuyucu.nextInt();

int[] koordinatlar = new int[engelSayisi];

i = 0; i < engelSayisi olduğu sürece

String engel = okuyucu.next();

j = 0; j < engel.length olduğu sürece

String değer;

Eğer engel.charAt(j) = ‘,’ değilse

Değer += engel.charAt(j);

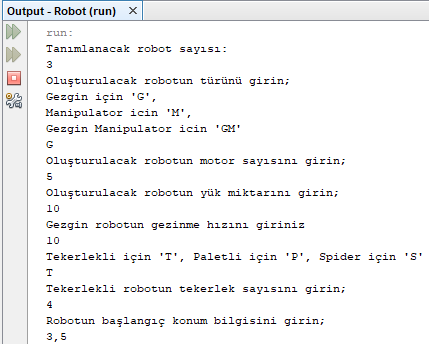
Koordinatlar[i] = (int)değer;

j’yi bir arttır başa dön

i’yi bir arttır başa dön

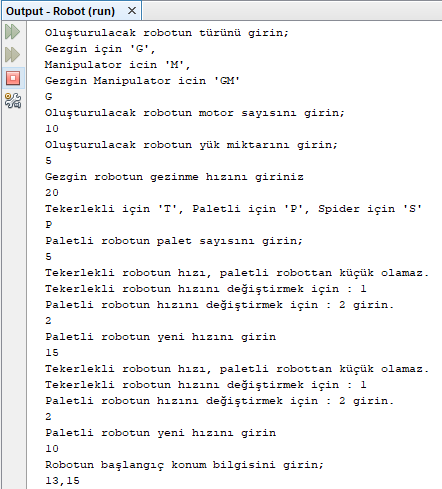
# Deneysel sonuçlar

# Tanımlanacak robot sayısı ve robot bilgilerinin kullanıcıdan alınması



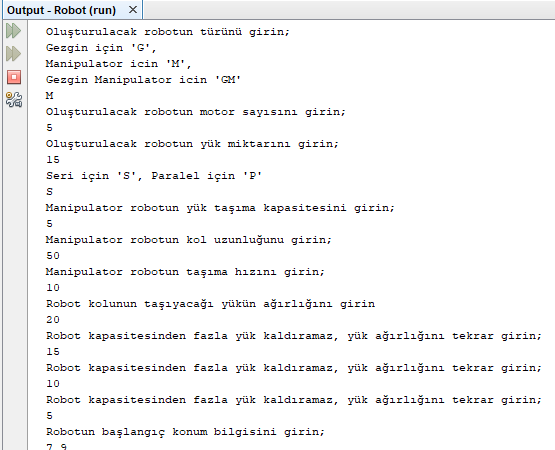
# *Şekil 1 Tanımlanacak robot sayısı ve robot bilgilerinin kullanıcıdan alınması*

# Gezgin robotlar için hız kıyaslamasının yapılması



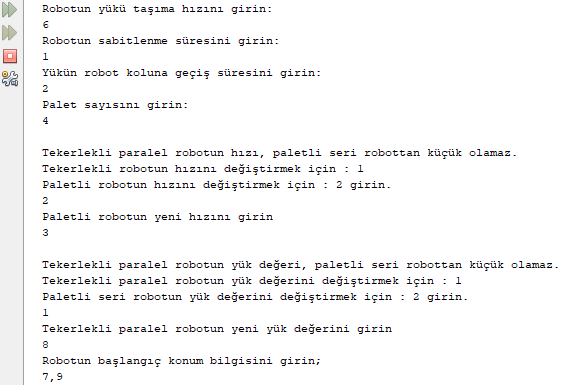
# *Şekil 2 Gezgin robotlar için hız kıyaslamasının* *yapılması*

# Manipülatör robotlar için kapasitesinden fazla yük taşıma kontrolü



# *Şekil 3 Manipülatör robotlar için kapasitesinden fazla yük taşıma kontrolü*

# Hibrit robotlarda gezgin kısmı için hız, manipülatör kısmı için yük kıyaslamasının yapılması



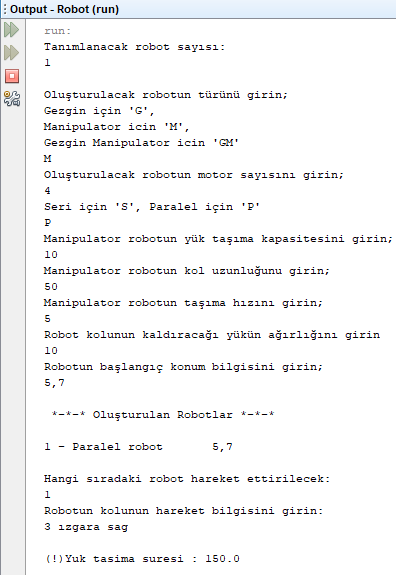
# *Şekil 4 Hibrit robotlarda gezgin kısmı için hız, manipülatör kısmı için yük kıyaslamasının yapılması*

# Gezgin veya hibrit robotlar için engel bilgisinin kullanıcıdan alınması

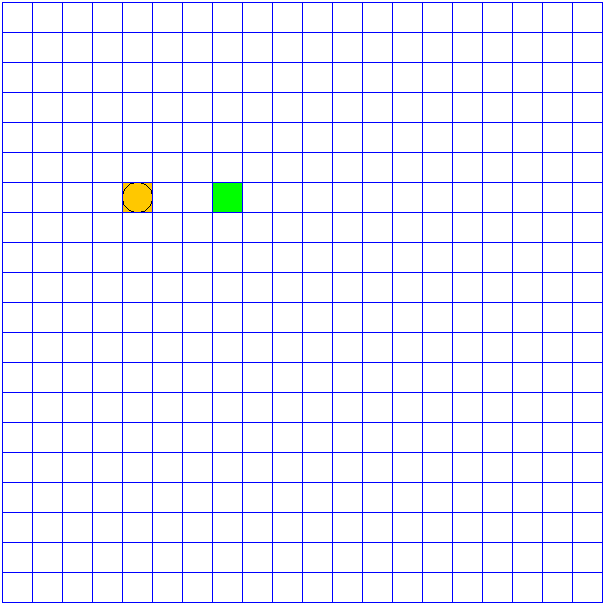
# *C:\Users\hacib\AppData\Local\Microsoft\Windows\INetCache\Content.Word\engel bilgisinin alınması.png*

# *Şekil 5 Gezgin veya hibrit robotlar için engel bilgisinin kullanıcıdan alınması*

# Manipülatör robotun kol uzunluğundan daha fazla uzaklığa yükü taşımaya çalışması durumu



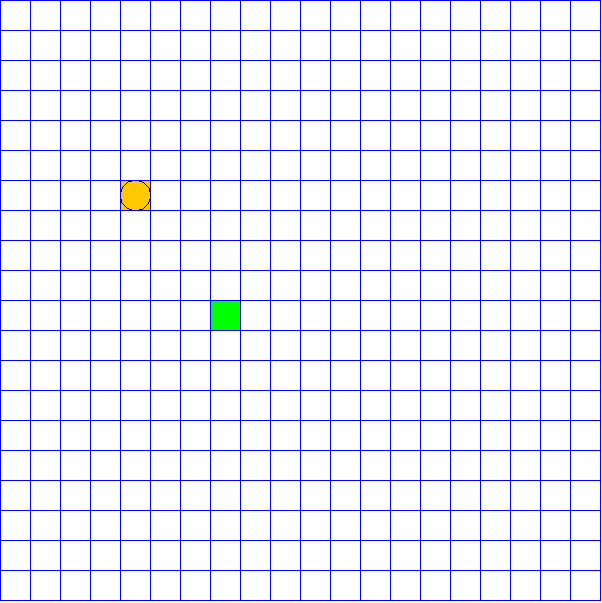
# *Şekil 6 Manipülatör robotun bilgilerinin alınması ve ilk hareketinin verilmesi*



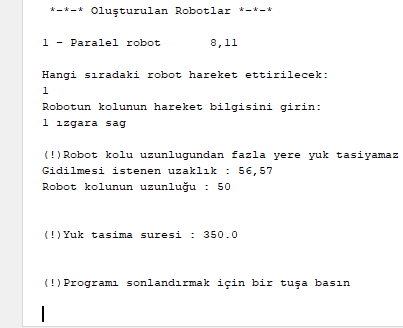
# *Şekil 7 Manipülatör robotun(turuncu kare) ve robot kolunun taşıdığı yükün(yeşil kare) grafik ekranında gösterilmesi*

# *C:\Users\hacib\AppData\Local\Microsoft\Windows\INetCache\Content.Word\yeni2.png*

# *Şekil 8 Manipülatör robotun koluna hareket bilgisinin verilmesi*

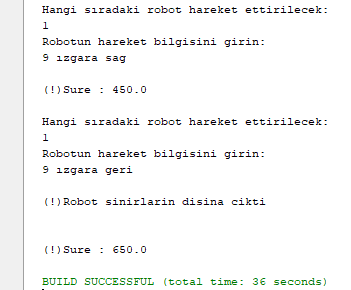


# *Şekil 9 Manipülatör robotun(turuncu kare) ve robot kolunun taşıdığı yükün(yeşil kare) grafik ekranında gösterilmesi*

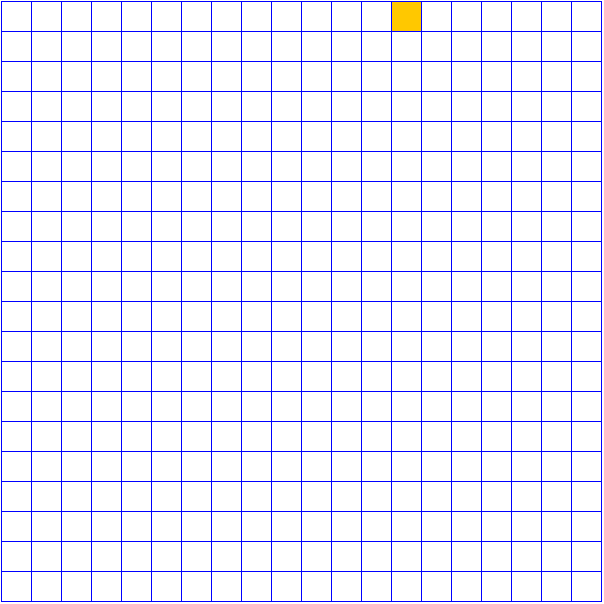


# *Şekil 10 Manipülatör robotun kolunun, kol uzunluğundan fazla yere yük taşımaya çalışması durumunun konsolda bildirilmesi*

# Robotun hareket edeceği alanda sınırların dışına çıkması durumu



# *Şekil 11 Robotun hareket edeceği alanda sınırların dışına çıkması durumunun konsolda bildirilmesi*



# *Şekil 12 Robotun hareket edeceği alanda sınırların dışına çıkması durumunda konsolda son konumunun gösterilmesi*

# Sonuçlar

Proje, kullanıcının istediği sayıda robot oluşturmasını, bu robotların belirlenen türlerden isteğine göre tanımlamasını ve ilk değer atamalarını sağlamaktadır.

Tanımlanan robotlar belirli koşullara göre üretilmektedir. Eğer kullanıcının girdiği değerler bu koşulları ihlal ederse, uygun değer girmesi için kullanıcıdan ilgili değer veya değerlerin tekrar belirlenmesi istenecektir. Aksi halde robotların diğer komutları icra etmesi mümkün olmayacaktır.

Tanımlanan robotlar ve engeller, JAVA grafik ekranında görüntülenir. Kullanıcının hareket ettirmek istediği robotu seçmesi istenir ve hareket ettirmesi istediği şekilde yön bilgilerini girmesi istenir.

Hareket eden robot gezgin robotsa veya hibrit robotun gezgin hareketini yaparken, eğer engele geldiyse ve engelden geçme özelliğine sahip değilse veya hareket edeceği alanın sınırlarının dışına çıktıysa, o zamana kadar geçen süre konsolda gösterilerek kullanıcı bilgilendirilir ve program sonlanır.

Hareket ettirilmesi istenen robot manipülatör robotsa, robot taşıyacağı yükü robot kolu vasıtası ile taşımaktadır. Robot kolunun belirli bir uzunluğu vardır ve eğer robot kolu yükü taşırken kol uzunluğunu aşarsa kullanıcı o zamana kadar geçen süre ve yapılan işlemin sonucu konsol ekranında gösterilerek bilgilendirilir ve program sonlandırılır.

# Kaynakça

[1] “Medium”

*Java OOP: Soyutlama — Abstraction — 4 – gokhanyavas – Medium*

17.04.2017

https://medium.com/gokhanyavas/java-oop-soyutlama-abstraction-4-ddddf8f820dd

29.11.2018

[2] “Medium”

*Java OOP: Miras Alma — Inheritance — 6*

17.04.2017

https://medium.com/gokhanyavas/java-oop-miras-alma-inheritance-6-95b0958f7dec

29.11.2018

[3] “Medium”

*Java OOP: Kapsülleme — Encapsulation — 5*

17.04.2017

https://medium.com/gokhanyavas/java-oop-kapsülleme-encapsulation-5-adfa594f7743 29.11.2018

[4] “Medium”

*Java OOP: Çok Biçimlilik — Polymorphism — 7 17.04.2017*

https://medium.com/gokhanyavas/java-oop-çok-biçimlilik-polymorphism-7-5f384c577b7f 29.11.2018

[5] “BilgisayarKavramlari”

*Swing Kütüphanesi – Bilgisayar Kavramlari*

01.06.2011

http://bilgisayarkavramlari.sadievrenseker.com/2011/06/01/swing-kutuphanesi/

02.12.2018

