YAZILIM YAŞAM DÖNGÜSÜ MODELLERİ (SOFTWARE LIFE-CYCLE MODELS):

YAZILIM YAŞAM DÖNGÜSÜ TEMEL AŞAMALARI:

Yazılım geliştirme yaşam döngüsü (Software Development Life Cycle “SDLC”) yazılım ürünü hem üretilirken hem de müşterinin kullanım süresi boyunca geçireceği tüm aşamaları kapsayan bir döngüdür. Yazılımı planlı bir şekilde geliştirebilmek için yazılım geliştirme sürecini zamanlamaya dayanarak ve içeriklerini de bölerek aşamalara ayırmalıyız. Yazılımın işlevleri, gereksinimleri sürekli olarak değiştiği ve geliştiği için yazılım yaşam döngüsünü tek yönlü (doğrusal) olarak düşünemeyiz. Birazdan bahsedeceğim aşamalar sürekli bir döngü içindedir. İhtiyaç olduğunda herhangi bir aşamadayken geriye dönebiliriz ve tekrar devam edebiliriz. Yazılım geliştirme aşamaları temel olarak şu şekildedir:

1. PLANLAMA:

Planlama aşaması yazılım geliştirme aşamalarının ilkidir. Müşterinin temel gereksinimleri belirlenir, yapacağımız proje için yapılabilirlik(fizibilite) çalışmaları yapılır, proje planlanır. Maliyetler tanımlanır. Sistemin yararları da tanımlanır.

1. ANALİZ:

Sistem gereksinimlerinin ve işlevlerinin detaylı olarak incelendiği aşamadır. Elimizdeki işler incelenir, temel sorunlar varsa ortaya çıkartılır. Bu aşamada amacımız sistemin kesin ihtiyaçlarını (gereksinimlerini) ve işlevlerini açığa kavuşturup belirli bir standartta belgelemek yani dokümante etmek içindir. Bu çalışmalar gruplarla yapılabilir. Müşteri, yazılım mühendisi, bilgisayar mühendisi, iş analisti, sistem analisti ,ürün yöneticisi gibi görevlilerin bir araya geldiği gruplarla yapılabilir. Eğer müşterinin gereksinimleri tam olarak belirli değilse yazılım/bilgisayar mühendisi ve müşteri arasında iletişimin daha fazla olması gerekir. Birlikte çalışabilirler. Çeşitli yazılım geliştirme metodolojilerinde analiz aşamasında dokümanlar ve test plan dokümanları oluşturulabilir.

1. TASARIM:

Gereksinimlerin tamamlanmasıyla sistemin tasarım aşamasına başlanabilir. Yazılım ürün tasarımı , müşterinin gereksinimlerini ve isteklerini karşılamak amacıyla yazılım ürünlerinin özelliklerinin, yapabileceklerinin ve arayüzlerinin belirlenmesi etkinliğidir. Tasarım aşamasında iki tür tasarımdan bahsedebiliriz. Bunlardan ilki ÜST SEVİYE VE MİMARİ TASARIM’dır. İkincisi ise DETAYLI TASARIM’dır. Mimari tasarım, önerilen sistemin yapısı (modüller, akış şemaları) ve organizasyon içerisindeki etkileşimleriyle ilgilenir. Bunun sonucunda mimari tasarım dokümanları oluşturulur. Detaylı tasarım aşamasında ise mimari tasarım dokümanları genelde yeniden gözden geçirilir, ele alınır.

Tasarım ve analiz aşamalarını ayırt etmek için “Problem nedir? / Problem Nasıl Çözülür? ” sorularını iyi kullanmak ile ilgilidir. Gereksinimlerin belirlendiği analiz aşaması problemin ne olduğu ile ilgilenir.

Şunu unutmayalım ki sistemdeki tüm problemler yazılım ürününün tamamlanmasıyla çözülemeyecektir.

1. GERÇEKLEŞTİRİM (KODLAMA VE TEST ETME) :

Tasarım aşamasının tamamlanmasıyla ya da belirli bir yere geldikten sonra kodlama aşaması başlar. Bu aşamada müşteriye teslim edilecek ürünü programlama (kodlama) aşamasıdır. Kaliteli kodlamaya kısaca değinecek olursak iyi kod , okunabilirliği ve bakımı kolay olan basit koddur . KISS ( KEEP IT SIMPLE) prensibine göre yazdığınız kodu yeni mezun olmuş birisine verdiğinizde ya da

1 ya da 2 gün içinde anlayabiliyorsa ve kodun üzerinde değişiklik yapabiliyorsa demek ki kodunuz iyi bir koddur. İsterseniz bir şirkette çalışın isterseniz bireysel projeler geliştirin mutlaka belirli bir kodlama kalite standartınız olsun. Bunlar neler olabilir: İsimlendirme standartları, tekrar eden kodlar, aşırı benzer işlevler, yorum satırı kullanımları, uzun metotlar, if-else blokları vb. dir.

Kodlama sürecinde ve kodlama sonrasında yapılan diğer önemli aşama da test aşamasıdır. Early testing (erken test et ) yaklaşımını kullanırsak , analiz aşamasından itibaren test bakış açısına sahip olmamız hata yapma oranımızı düşürecektir. Aynı zamanda maliyetleri( zaman, para gibi ) de düşürecektir. Sürece ve duruma göre uygulanabilecek çok farklı kategoride ve derinlikte test türü bulunmaktadır.

1. TESLİM VE BAKIM:

Tüm test aşamaları tamamlanmıştır. Yazılım ürünü sahaya teslim edilebilir bir versiyonu çıkartılır ve teslim aşaması da gerçekleştirilmiş olur. Teslim çıktısı olan ürün tek başına yeterli değildir. Mutlaka son kullanıcılar için kullanım kılavuzu aynı zamanda versiyon fark dokümanı da oluşturulmuş olmalıdır. Teslim aşamasıyla birlikte bakım aşaması da başlar. Farklı bakım faaliyetleri vardır. Eğer bir hata çıkarsa hata giderici , önleyici, altyapıyı iyileştirici , ürüne yeni bir özellik eklemek gibi bakım faaliyetleri mevcuttur.

Bahsettiğim yaşam döngüsünün temel adımlarına çekirdek süreçler (core processes) de denilebilir. Bu süreçleri gerçekleştirebilmek için Yazılım Belirtim Yöntemleri ve Yazılım Süreç Modelleri kullanılmaktadır.

YAZILIM SÜREÇ MODELLERİ:

Yazılım yaşam döngüsünde belirtilen süreçlerin geliştirme aşamasında, hangi düzen veya sırada, nasıl uygulamamız gerektiğini tanımlayan modellerdir. Yazılım sürecindeki karmaşıklığı azaltır ve krizi önler. Ürünlerin beklenen kalitede olması için süreçleri kontrol etmek gerekir. Yazılım süreç modellerinden belli başlı olanları aşağıda verilmiştir:

1.KODLA VE DÜZELT (CODE AND FİX):

2.ŞELALE MODELİ (WATERFALL MODEL):

3.V MODELİ (V-SHAPED MODEL):

4.EVRİMSEL GELİŞTİRME (EVOLUTIONARY DEVELOPMENT):

5.PROTOTİPLEME (PROTOTYPING):

6.SPİRAL MODEL:

7.FORMAL SİSTEM GELİŞTİRME (FORMAL SYSTEM DEVELOPMENT):

8.YENİDEN KULLANIMA YÖNELİK GELİŞTİRME (RE-USE BASED DEVELOPMENT):

9.ARTIRIMLI GELİŞTİRME (INCREMENTAL DEVELOPMENT):

10.BİRLEŞİK SÜREÇ (UNIFIED PROCESS):

11.ÇEVİK MODELLER (AGILE PROGRAMMING: XP, SCRUM):

1.KODLA VE DÜZELT (CODE AND FİX):

Bu model genellikle resmi olmayan bir ürün fikriyle başlar ve program ürün hazır olana kadar ya da gerekli zaman bitene kadar kodlama yapılarak devam eder.

Avantajları şöyledir:

1. Herhangi bir planlamaya ihtiyaç duyulmaz.

2. Çok küçük projelerde veya kısa ömürlü prototiplerde uygulanabilir.

3. Program aşamaları çabuk geçirilir.

4. Uzman görüşüne ihtiyaç düşüktür , herkes bu modeli kullanılabilir.

Dezavantajları şöyledir:

1. Kontrollü değildir.
2. Kaynak planlaması yoktur.
3. Bitiş süresi belli değildir.
4. Hataların bulunması ve doğrulaması zordur.
5. Kodları düzeltmek maliyetli olabilir.
6. Kodlar kullanıcının ihtiyacını karşılamayabilir.
7. Kodlar sonradan değiştirmek için planlanmadığından esnek değildir, değiştirilmesi zordur.

2.ÇAĞLAYAN MODELİ:

Şelale yönteminde yazılım geliştirme süreci analiz, tasarım , kodlama, test , sürüm ve bakım gibi aşamalardan oluşur. Geleneksel yazılım metotlarında bu safhalar şelale modelinde olduğu gibi lineer olarak işler. Her safha, başlangıç noktasında bir önceki safhanın ürettiklerini bulur. Kendi bünyesindeki değişiklikler doğrultusunda teslim aldıklarını bir sonraki safhanın kullanabileceği şekilde değiştirir.

Avantajları:

1. Kullanımı ve anlaması basittir.
2. Yöntemi kolaydır.
3. Projenin safhaları ayrı olduğundan iş bölümü ve planı projenin başında net bir şekilde bellidir. Bu durum projenin yönetimini de oldukça kolay hale getirir.
4. Şelale modeli çok küçük ve gereksinimleri çok iyi anlaşılmış projelerde iyi çalışır.

Dezavantajları:

1. Karmaşık ve nesne yönelimli projeler için uygun değildir.
2. Devam eden ve uzun projeler için zayıftır.
3. Projede oluşabilecek her türlü değişime elverişsiz, katı bir modeldir. Yapılan her değişiklik maliyeti büyük oranda artırır.
4. Müşteri memnuniyetini sağlamak çok zordur çünkü gelişim ve değişime açık bir model değildir.
5. Model safhalardan oluştuğu için ürün son safhada tamamlanır, gereksinimlerin iyi tanımlanmadığı müşterinin ne istediğinin anlaşılmadığı bir projede bu durum projenin bittikten sonra iptal edilmesine ve başka gerginliklere sebep olmaktadır.

3. V MODELİ:

V- model (yazılım geliştirme) şelale ( waterfall ) modelinin gelişmiş hali olarak düşünülebilecek bir yazılım geliştirme süreci sunar. Doğrusal bir yönde ilerlemek yerine, süreç adımları kodlama evresinden sonra yukarıya doğru eğim alır ve tipik V şeklini oluşturur. V- model geliştirme yaşam çevriminin her bir evresi arasındaki ilişkileri gösterir.

Avantajları:

1. Doğrulama ve onaylama planları erken aşamalarda vurgulanır.
2. Doğrulama ve onaylama sadece son üründe değil tüm teslim edilebilir ürünlerde uygulanır.
3. Proje yönetimi tarafında takibi kolaydır.
4. Kullanımı kolaydır.

Dezavantajları:

1. Aynı zamanda gerçekleştirilebilecek olaylara kolay imkan tanımaz.
2. Fazlar arasında tekrarlamaları kullanmaz.
3. Risk çözümleme ile ilgili aktiviteleri içermez.
4. Yazılım da diğer sistemler gibi zamanla evrimleşir.
5. Geliştirme devam ettikçe iş ve ürün gereksinimleri de değişkenlik gösterebilir.
6. Son ürüne ulaşma düz bir çizgi ile ifade edilemez.

4.EVRİMSEL GELİŞTİRME (EVOLUTIONARY DEVELOPMENT):

İlk tam ölçekli modeldir. Coğrafik olarak geniş alana yayılmış, çok birimli organizasyonlar için önerilmektedir (Banka uygulamaları). Her aşamada üretilen ürünler , üretildikleri alan için tam işlevselliği içermektedirler. Pilot uygulama kullan, test et, güncelle diğer birimlere taşı. Modelin başarısı ilk evrimin başarısına bağlıdır.

Avantajları:

1. Kullanıcıların kendi gereksinimlerini daha iyi anlamalarını sağlar.
2. Sürekli değerlendirme erken aşamalardaki geliştirme risklerini azaltır.
3. Hatalar azalır.

Dezavantajları:

1. Sürecin görünürlüğü azdır. (Düzenli teslim edilebilir ürün yoktur.)
2. Sistemler sıklıkla iyi yapılandırılmaz. (Sürekli değişiklik yazılımın yapısına zarar verir)
3. Bakımı zordur.
4. Yazılım gereksinimini yenilemek gerekebilir.

5.PROTOTİPLEME:

Gereksinimler hızlıca toplanarak işe başlanılır. Geliştiriciler ve kullanıcılar aynı masa etrafında buluşarak yazılımdan elde edilecek bütün çıktılara, bu çıktılar için gerekli girdilerin nasıl sağlanacağına, nasıl korunacağına, hangi işlemlere uğrayacağına karar verirler. ​​ Daha sonra hızlıca yapılan bir tasarım ile yazılımın kullanıcıya yansıyacak yönünü aktaran bir​​ ilk örnek​​ üretilir. Prototip kullanıcının kullanımına ve değerlendirilmesine sunulur. Bu değerlendirmelere bakılarak​​ ilk örnek​​ üzerinde gerekli​​ değişiklikler yapılır. Prototipin yeni hali kullanıcı tarafından yeniden değerlendirilir. Böylece kullanıcının istediği yazılıma iyice yaklaşılmış bir​​ ilk örnek​​ üzerinde yazılımın​​ neler yapacağı konusunda kullanıcı ile anlaşmaya varılır.​​ Doğrusal modelin döngüsel versiyonudur.​​ Bu modelde, gereksinim analizi ve prototipleme için tasarım yapıldıktan​​ sonra, geliştirme​​ süreci başlatılır.​​ Prototipleme yaratıldıktan sonra, müşteriye değerlendirme için verilir.​​ Müşteri paketi test eder ve düşüncelerini, ürünü müşterinin tam beklentilerine göre düzenleyen geliştiriciye iletir.​​ Sınırlı sayıdaki yinelemelerden sonra, son yazılım paketi müşteriye verilir.​​ Bu metodolojide, yazılım müşteri ve geliştirici arasında periyodik bilgi gidip gelmeleri sonucunda gelişir.

Avantajları:

1. Kullanıcı sistem gereksinimlerini görebilir.
2. Karmaşa ve yanlış anlaşılmaları engeller.
3. Yeni ve beklenmeyen gereksinimler netleştirilebilir.
4. Risk kontrolü sağlanır.

Dezavantajları:

1. Belgelendirmesi olmayan hızlı ve kirli (quick and dirty ) prototipler vardır.
2. Düzeltme aşaması atlanırsa , düşük performansa yol açar.
3. Müşteri prototipten de son ürün gibi görünüm ve etki bekler.

6.SPİRAL MODEL:

Sarmal modeli aynı safhalara geri dönülmesinin bir zorunluluk olduğunu vurgular, sarmal modeli şelale modelinde yok sayılan riskleri göz önünde bulundurur. Proje çevrimlere ayrılır ve her bir çevrimin riskleri ayrı ayrı ele alınır. Çağdaş modellere son derece yakındır. ​​

Planlama:

Üretilecek ara ürün için planlama, amaç belirleme, bir önceki adımda üretilen ara ürün ile bütünleştirmeler yapılır.

Risk analizi:

Risk seçeneklerinin araştırılması ve risklerin belirlenmesi

Üretim:

Ara ürünün üretilmesi

Kullanıcı değerlendirmesi:

Ara ürün ile ilgili olarak kullanıcı tarafından yapılan sınama ve değerlendirmeler

Risk analizi olgusu ön plana çıkmıştır. Her döngü bir fazı ifade eder.

Avantajları:

1. Kullanıcılar sistemi erken görebilir.
2. Geliştirmeyi küçük parçalara böler. En riskli kısımlar önce gerçekleştirilir.
3. Pek çok yazılım modelini içinde bulundurur.
4. Riske duyarlı yaklaşımı potansiyel zorlukları engeller.
5. Seçeneklere erken dikkate odaklanır.
6. Hataları erken gidermeye odaklıdır.
7. Yazılım-donanım sistemi geliştirme için bir çerçeve sağlar.

Dezavantajları:

1. Küçük ve düşük riskli projeler için pahalı bir yöntemdir.
2. Komplekstir. (karmaşıktır)
3. Spiral sonsuza gidebilir.
4. Ara adımların fazlalığı nedeniyle çok fazla dokümantasyon gerektirir.

7.FORMAL SİSTEM GELİŞTİRME (FORMAL SYSTEM DEVELOPMENT):

Formal Sistem Geliştirme Modeli​​ yazılım tasarım ve gerçekleştirmesiyle ilgili matematiksel bir tekniktir. Bu modelin temelinde karmaşık sistemleri geliştirme ve program geliştirmeye destek yatar. Formal Sistem Geliştirme Metodu​​ kullanıcı sistemi kullanmaya başladığında karşısına çıkan belirtim hatalarını minimize eder. Formal belirtim, tasarım ve geçerleme kullanarak yazılımda doğruluğun geliştirilmesini vurgular.​​ Yazılım artımlarla geliştirilir.​​ Sürekli tümleştirme vardır ve fonksiyonellik tümleştirilen yazılım artımları ile artar.​​ Felsefesi pahalı hata ayıklama işlemini engellemek için kodu ilk yazarken doğru yazmak ve test aşamasından doğruluğunu sağlamaktır.

Avantajları:

1. Yazılımdaki belirsizlikleri, eksiklikleri ve uyumsuzlukları saptar.
2. Hatasız yazılım geliştirme imkanları sunar.
3. Her iterasyondan (yinelemeden) sonra aşamalı olarak artan efektif çözümler sunar.
4. Karmaşık değildir.

Dezavantajları:

1. Çok zaman alan ve pahalı bir yöntemdir.
2. Kullanımı esnasında teknik olmayan personelle iletişim mekanizması zor işler.​​ ​​
3. Sadece birkaç geliştirici bu modelin uygulamasıyla ilgili temel bilgilere sahip olması için yaygın eğitim gerektirir.

8.YENİDEN KULLANIMA YÖNELİK GELİŞTİRME (RE-USE BASED DEVELOPMENT):

Organizasyon tarafından daha önce hazırlanmış veya dışarıdan temin edilmiş yazılımların (veya yazılım parçalarının) kullanılması ile geliştirme yapılması son yıllarda popülaritesi artan bir yaklaşımdır. Organizasyonların olgunlukları arttıkça, bu tür uygulamalar yapabilmek için alt yapı kurulmuş olmaktadır.

Avantajları:

1. Kaynak kontrolü mümkündür.
2. Maliyet denetimi yapmak mümkündür.
3. Basit ve anlaşılırdır.
4. Önceden oluşturulmuş sınıflar tekrardan kullanılabilir.
5. Kısa sürede yazılım geliştirilebilir.

Dezavantajları:

1. Gereksinimleri anlamak güçtür.
2. Pahalıdır.
3. Uzmanlık gerektirir.
4. Başarım garantisi yoktur.

9.ARTIMLI GELİŞTİRME (INCREMENTAL DEVELOPMENT):

Eğer bir müşterinin ürünlerinde değişikliğe ihtiyaçları varsa, artımlı model ihtiyaç olan bu değişikliğe ayak uydurur. Artırımsal model bir takvime bağlı olarak yazılımı kesim kesim geliştirip teslim etmeye dayanır. Her bir yeni kesim öncekinin üstüne bazı ek işlevlerin eklenmesini öngörür. Artırımsal model yazılım geliştirmenin kısıtlı sayıda çalışanla işin yapılmasını sağlama gibi bir üstünlüğü vardır. Ayrıca çalışanlar da her artırım geçildiğinde uygulama alanına ilişkin daha çok deneyim kazanmış olurlar. Bu modelde bir taraftan üretim bir taraftan da kullanım yapılır. Önceki modellerde ürünlerdeki değişiklikler göz önünde bulundurulmaz. Bu model doğal olarak yinelemelidir.​​ Yeniden kullanılabilir bir ürün,​​ fonksiyonellik sağlamış bir şekilde tüm döngülerin sonunda ortaya çıkar.

Avantajları:

1. Sistem için gerekli olan gereksinimler müşterilerle belirlenir.
2. Gereksinimlerin önemine göre teslim edilecek artımlar belirlenir.
3. Öncelikle en önemli gereksinimleri karşılayan çekirdek bir sistem geliştirilir.
4. Erken artımlar prototip gibi davranarak, gereksinimlerin daha iyi anlaşılmasını sağlar.
5. Tüm projenin başarısız olma riskini azaltır.
6. En önemli sistem özellikleri daha fazla sınanma (test edilme) imkanı bulmuş olur.

Dezavantajları:

1. Artımları tanımlamak için tüm sistemlerin tanımlanmasına ihtiyaç vardır.
2. Gereksinimleri doğru boyuttaki artımlara atamak bazen zor olabilir.
3. Deneyimli personel gerektirir.

10.BİRLEŞİK SÜREÇ (UNIFIED PROCESS):

Nesneye dayalı yazılım geliştirmek için var olan yöntemlerin deneyimler sonucu kabul gören en iyi özellikleri bir araya getirilerek tümleştirilmiş yazılım geliştirme süreci (The Unified Process - UP) oluşturulmuştur. Yinelemeli, artırmalı ve evrimseldir aynı zamanda da risk güdümlüdür.

Avantajları:

1. Değişen isteklere uygunluk
2. Erken geri besleme
3. Büyük sistemlerde çözme kolaylığı
4. Her iterasyonda deneyim kazanılması
5. Risklerin erken giderilmesi
6. Erken ürün elde etme ve takımda moral yükselmesi

Dezavantajları:

1. Risk yönetimi zayıftır.
2. Dokümantasyon yükü ağır olduğundan pahalı olabilir.
3. Karmaşıktır.

11.ÇEVİK MODELLER (AGILE PROGRAMMING):

İteratif geliştirme temelli bir grup yazılım geliştirme metodolojisine dayanır. Burada gereksinimler ve çözümler kendinden örgütlü olan farklı grupların ortak çalışmasıyla olgunlaşır. Agile metotları genellikle sık denetim ve adaptasyonun teşvik edildiği disipline edilmiş bir proje yönetim sürecini, takım çalışmasının, kendinden örgütlenmenin ve izlenebilirliğin özendirildiği liderlik felsefesini, kaliteli yazılımların hızlı biçimde geliştirilmesinin hedeflendiği en iyi mühendislik uygulamalarını ve yazılım geliştirme ile müşteri ihtiyaçlarını ve firma amaçlarını yan yana getiren business yaklaşımı destekler.​​ Agile metotları bir yazılım geliştirme yaklaşımı değil, geliştirme süreçleri topluluğudur.​​

Çevik yazılım (Agile Development) bir yandan bir değer sistemini, diğer yandan da somut yazılım metotlarını içerir. Çevik yazılıma, yazılım sektöründe yeni bir filozofi akımı, ya da yeni bir yazılım meta modeli olarak bakabiliriz. Bu yeni filozofinin somut örnekleri arasında Extreme Programming (XP), Scrum ve Lean Development bulunmaktadır.

Prensipler:

1. Teslimatı hızlı ve aralıklı yaparak müşteri memnuniyeti sağlamak
2. Çalışan yazılımın aylar yerine haftalar süren zamanda teslim etmek
3. Sürecin esas değerlendirme kriteri çalışan sistemdir.
4. Gereksinimlere sonradan eklenen değişiklikler mutlaka dikkate alınır.
5. Business ve teknik ekip arasında sıkı ve günlük çalışma.
6. En iyi iletişim yolu yüz yüze görüşme kabul edilir.
7. Proje güvenilir ve motivasyonu yüksek kişiler üzerine kurulur
8. Teknik başarı ve iyi tasarıma düzenli iltifat
9. Basitlik
10. Kendinden örgütlü takımlar
11. Değişen koşullara adaptasyon

Avantajları:

1. İnsanın doğal eğilimine çok yatkındır. Öğrenim gerektirmez, adaptasyon hızlıdır.
2. Kısa döngüler dolayısıyla takım elemanlarında motivasyon çok yüksektir. Verim artışı sağlanır.
3. Sık çıktı üretip geri besleme aldığından kaynağı, müşteri ihtiyaçlarına ve sonuca kanalize etmeye odaklanır.
4. Plan aşamasında ayrıntılı plan yerine iterasyonun planı yapılır.
5. Değişime açıklık ve esneklik en üst düzeydedir.
6. Takım oyunu

Dezavantajları:

1. Kurumsal bir yapıda uygulaması gerçekten zor.
2. Dokümantasyon hakkındaki taşları yerinden oynatan bir yaklaşımdır.
3. Sürekli değişen ihtiyaçlar dolayısıyla aşırı çalışma
4. Takım üzerindeki hedef baskısı

Ad Soyad: Nur Şenkara

Öğrenci No: 180601034