MALATYA TURGUT

ÖZAL ÜNİVERSİTESİ

MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ

Yazılım Mühendisliği Bölümü

YAZM354-Paralel

Programlama Dersi

Vize Ödevi

Hazırlayan: Damla KAYNARCA

Sınıfı:3.sınıf

No:02200201019

Soru1:Paralel programlama nedir?

Cevap1:

*Aynı anda birden fazla prosesin birlikte çalışma mekanizmasına paralel programlama denir.

* İşlemcinin aynı anda birden fazla işlem yapabilme yeteneğine denir.
* Bir örnekle izah etmek gerekirse;
* Bilgisayar üzerinde bir meet toplantısı gerçekleştirdiğimizi farz edelim .Sunumu yapan kişinin hem ekranda yansıyan sayfalarının görüntüsünü görebiliyoruz hem de mikrofonundan ses duyabiliyoruz işte tıpkı bu örnekte olduğu gibi aynı anda birden fazla iş yapabilme yeteneğine diyoruz.
* Bir başka örnek ;
* Aynı anda birden fazla prosesin (kullanımda olan bir bilgisayar üzerinde hem internet explorer hem bir servis hem de bir editörün açık olduğunu düşünelim.)buradaki çalışma mekanizmasıdır.
* Bu proseslerin aynı anda birlikte çalışma yapısına paralel programlama diyoruz.
* Konuyu biraz detaylandıracak olursak bigisayarınızda Task manager--->Performance--->Cpu kısmında kaynak noktasında i7 işlemcili bilgisayarımda (7-1)=6 adet core yapımız vardır ve 6*2=12 logical prosess yapısı paralel olarak çalışma yapmamızı sağlayacaktır.

* Karmaşık problemler, karmaşık çözümler gerektirir. Bir programın çalışmasının bitmesi için saatlerce beklemek yerine neden paralel programlama kullanmıyoruz?

* Paralel programlama, geliştiricilerin bir programın tamamlaması gereken görevleri paralel olarak yapılabilecek daha küçük iş bölümlerine ayırmasına yardımcı olur.
* Paralel programlama, geliştiricilerin verimli paralel algoritmalar ve kodlar oluşturması için önceden daha fazla zaman harcayan bir çaba olabilirken, programı aynı anda birden fazla hesaplama düğümü ve CPU çekirdeği üzerinde çalıştırarak paralel işlem gücünden yararlanarak genel olarak zaman kazandırır.

Soru2:Paralel programlamanın avantajları nelerdir?

Cevap2:

* Paralel bilgi işlem, uygulamaların daha kısa bir duvar saati süresinde yürütülmesine izin vererek zamandan tasarruf sağlar.
* Daha Büyük Sorunları Kısa Sürede Çözün.
* Seri bilgi işlemle karşılaştırıldığında, paralel bilgi işlem karmaşık, gerçek dünya olaylarını modellemek, simüle etmek ve anlamak için çok daha uygundur.
* Bir göreve daha fazla kaynak ayırmak, potansiyel maliyet tasarrufuyla birlikte tamamlanma süresini kısaltacaktır. Paralel bilgisayarlar, ucuz, ticari bileşenlerden oluşturulabilir.
* Pek çok sorun o kadar büyük ve/veya karmaşıktır ki, özellikle sınırlı bilgisayar belleği söz konusu olduğunda, bunları tek bir bilgisayarda çözmek pratik değildir veya imkansızdır.
* Birden çok bilgi işlem kaynağını kullanarak birçok şeyi aynı anda yapabilirsiniz.
* Geniş Alan Ağında (WAN) veya hatta internette bilgisayar kaynaklarını kullanabilir.

* Organize olmanıza yardımcı olabilir. İnternetiniz varsa, iletişim ve sosyal ağlar daha kolay hale getirilebilir.
* Büyük veri depolama ve hızlı veri hesaplamalarına sahiptir.
* En genel özetiyle;
* Projeleri programladığımızda hızlı bir şekilde çalışmamıza imkan verir.
* Sistemimizde bulunan kaynakların ölçeklenebilirliğini sağlar.
* Eş zamanlı çalışma imkanı sağlar.

Soru3:Paralel programlama dezavantajları nelerdir?

Cevap3:

* Paralel mimariyi hedeflemek için programlama biraz zordur, ancak doğru anlayış ve pratikle, gitmeye hazırsınız.
* Paralel bilgi işlemin kullanılması, çok çekirdekli işlemciler kullanarak hesaplamalı ve veri yoğun sorunları çözmenize olanak tanır, ancak bazen bu, kontrol algoritmamızın bazılarını etkiler ve iyi sonuçlar vermez ve bu, paralel seçeneği nedeniyle sistemin yakınsamasını da etkileyebilir. .
* Ortaya çıkan ekstra maliyet (yani artan yürütme süresi), veri aktarımları, senkronizasyon, iletişim, iş parçacığı oluşturma/imha vb. nedeniyledir. Bu maliyetler bazen oldukça büyük olabilir ve aslında paralelleştirmeden kaynaklanan kazançları aşabilir.
* Gelişmiş performans için farklı hedef mimariler için çeşitli kod ince ayarları yapılmalıdır.
* Kümeler durumunda daha iyi soğutma teknolojileri gereklidir.
* Güç tüketimi, çok çekirdekli mimariler tarafından çok fazladır.

* Paralel çözümlerin uygulanması daha zordur, hata ayıklamak veya doğru olduklarını kanıtlamak daha zordur ve iletişim ve koordinasyon ek yükü nedeniyle genellikle seri emsallerinden daha kötü performans gösterirler.

Soru4:Synchronized nelerdir?

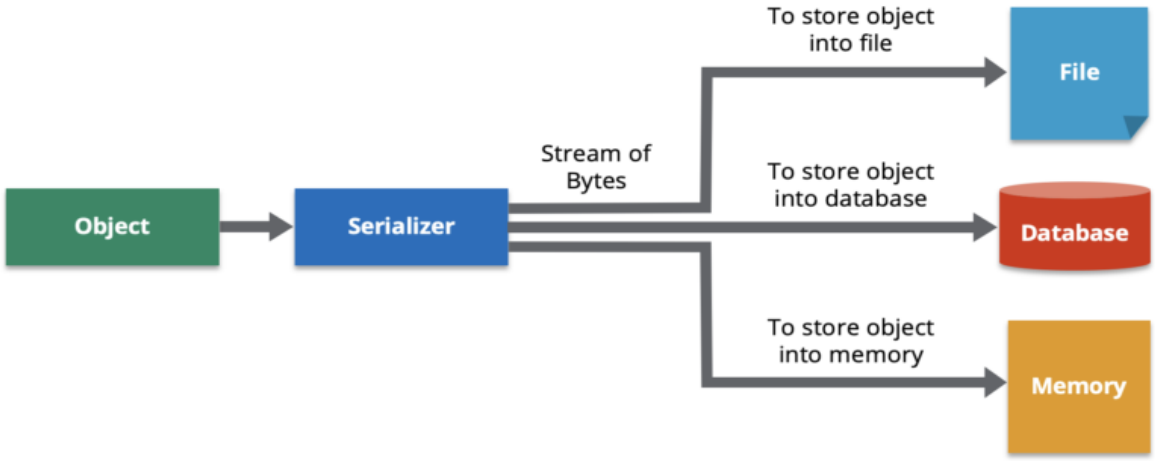
Cevap4:

* En açık ifadeyle synchronized yapısı bir işlem bitmeden başka bir işlemin girmesine izin vermiyor.Buna Synchorized yapısı diyoruz.
* Çok threadli bir uygulama söz konusu olduğu zaman threadlerin aktivitelerini kontrol etmek gerekebilir. Bazı durumlarda iki yada daha fazla thread paylaşılan bir kaynağa aynı anda erişmek ve üzerinde değişiklik yapmak isteyebilir. Örneğin bir threadin bir dosyaya yazma işlemi yaptığı sırada bir başka threadin de aynı işlemi yapmak istemesi gibi. Böyle durumlarda kaynağa ilk ulaşan threadin işini tamamlayıncaya kadar ilgili kaynağın lock dediğimiz kilit mekanizması ile erişime kapatılması daha sonra ise tekrar erişime açılması gerekir. Java programlama dilinde her obje bu lock mekanizması ile koruma altına alınabilir ve bu işlem synchronized ifadesi ile sağlanır.

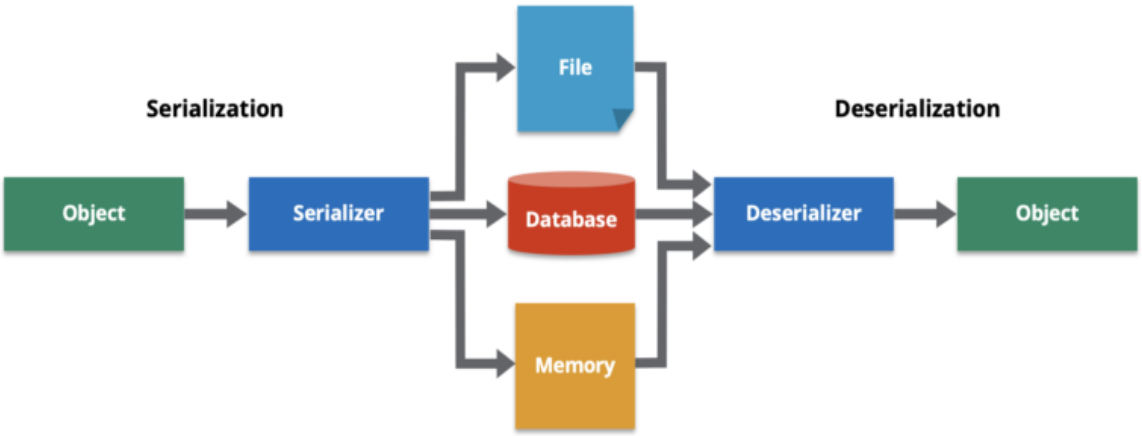
Soru5:Serileştirme nedir?

Cevap5:

*Seri hale getirme , bir veri nesnesini (veri depolama bölgesinde temsil edilen kod ve verilerin birleşimi), nesnenin durumunu kolayca iletilebilir bir biçimde kaydeden bir bayt dizisine dönüştürme işlemidir. Bu seri hale getirilmiş formda, veriler başka bir veri deposuna (bellek içi bilgi işlem platformu gibi), uygulamaya veya başka bir hedefe iletilebilir .



Veri serileştirme, bir nesneyi daha kolay kaydetmek veya iletmek için bir bayt akışına dönüştürme işlemidir.Bir bayt serisinden bir veri yapısı veya nesne oluşturmak olan ters işlem, **seri hale getirmedir** . Seri durumdan çıkarma işlemi, nesneyi yeniden oluşturur, böylece verilerin bir programlama dilinde yerel bir yapı olarak okunmasını ve değiştirilmesini kolaylaştırır.



* Seri hale getirme ve serisini kaldırma, veri nesnelerini taşınabilir bir biçime dönüştürmek/yeniden oluşturmak için birlikte çalışır.
* Serileştirme, bir nesnenin durumunu kaydetmemizi ve nesneyi yeni bir konumda yeniden oluşturmamızı sağlar. Serileştirme, hem nesnenin depolanmasını hem de veri alışverişini kapsar.

Nesneler birkaç bileşenden oluştuğundan, tüm parçaları kaydetmek veya teslim etmek tipik olarak önemli bir kodlama çabası gerektirir, bu nedenle serileştirme, nesneyi paylaşılabilir bir biçimde yakalamanın standart bir yoludur. Serileştirme ile nesneleri aktarabiliriz:

* Mesajlaşma kullanım durumları için kablo üzerinden
* REST API'leri gibi web hizmetleri aracılığıyla uygulamadan uygulamaya
* Güvenlik duvarları aracılığıyla (JSON veya XML dizeleri olarak)
* Etki alanları arasında
* Diğer veri depolarına
* Verilerdeki zaman içindeki değişiklikleri belirlemek için
* Uygulamalar arasında güvenlik ve kullanıcıya özel ayrıntıları onurlandırırken

Dağıtık Sistemler İçin Veri Serileştirme Neden Önemlidir?

* Bazı dağıtılmış sistemlerde, veriler ve kopyaları birden çok küme üyesi üzerinde farklı bölümlerde depolanır. Yerel üyede veri yoksa, sistem bu verileri başka bir üyeden alır. Bu, aşağıdaki gibi kullanım durumları için serileştirme gerektirir:
* Bir haritaya **anahtar/değer** nesneleri ekleme
* Öğeleri sıraya, kümeye veya listeye koyma
* Bir lambda işlevini başka bir sunucuya gönderme
* Bir harita içindeki bir girişi işleme

* Bir nesneyi kilitleme
* Bir konuya mesaj gönderme

Soru 6:Thread Lock nedir?

Cevap6:

Lock(kilit) mantığı basitçe, bir thread' in kaynağa erişmeden önce o kaynağı kilitleyerek, başka hiç bir thread' in o kaynağı kullanmasına izin vermemesidir. … İşini bitiren thread, kaynak üzerindeki kilidi serbest bırakarak, kaynağı diğer thread' lerin kullanımına bırakır.

Soru 7:Thread Pool Executer Services Nedir?

Cevap7:

* **İş Parçacığı Oluşturma** : Uygulamanızın görevleri aynı anda çalıştırmak için kullanabileceği iş parçacığı oluşturmak için çeşitli yöntemler, daha özel olarak bir iş parçacığı havuzu sağlar.
* **Thread Management** : Thread havuzundaki threadlerin yaşam döngüsünü yönetir. Yürütülmek üzere bir görev göndermeden önce iş parçacığı havuzundaki iş parçacıklarının etkin mi, meşgul mü, yoksa ölü mü olduğu konusunda endişelenmenize gerek yok.
* **Görev gönderme ve yürütme** : Executors çerçevesi, görevlerin iş parçacığı havuzunda yürütülmek üzere gönderilmesi için yöntemler sağlar ve ayrıca size görevlerin ne zaman yürütüleceğine karar verme gücü verir. Örneğin, bir görevi şimdi yürütülmesi için gönderebilir veya daha sonra yürütülmesi için planlayabilir veya periyodik olarak yürütülmesini sağlayabilirsiniz.

* Java Concurrency API, iş parçacıkları oluşturmak ve yönetmek için gereken her şeyi kapsayan aşağıdaki üç yürütücü arabirimini tanımlar -
* **Yürütücü**execute() - Bir nesne tarafından belirtilen bir görevi başlatmak için çağrılan bir yöntemi içeren basit bir arabirim Runnable.
* **ExecutorService** - Bunun bir alt arabirimi, Executorgörevlerin yaşam döngüsünü yönetmek için işlevsellik ekler. submit()Ayrıca , aşırı yüklenmiş sürümleri Runnablebir nesnenin yanı sıra a'yı da kabul edebilen bir yöntem sağlar Callable. Callable nesneleri, Callable nesnesi tarafından belirtilen görevin de bir değer döndürebilmesi dışında Runnable'a benzer.**ScheduledExecutorService** - ExecutorService. Görevlerin yürütülmesini zamanlamak için işlevsellik ekler.
* API, yukarıdaki üç arayüzün yanı sıra, farklı türde yürütücü hizmetleri oluşturmak için fabrika yöntemlerini içeren bir
* Executors sınıfı da sağlar.

Soru 8:await nedir?

Cevap8:

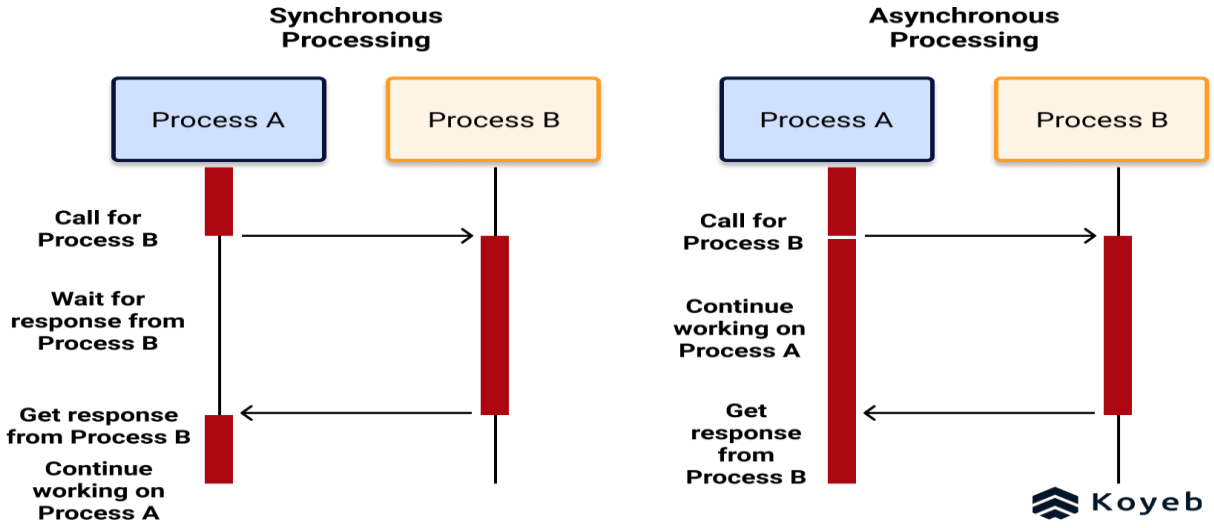
* Mevcut iş parçacığının, iş parçacığı kesintiye uğramadıkça veya belirtilen bekleme süresi geçmedikçe, sayımsıfıra geri sayana kadar beklemesine neden olur. Geçerli sayım sıfırsa, bu yöntem hemen true değeriyle döner. Geçerli sayım sıfırdan büyükse, mevcut iş parçacığı iş parçacığı programlama amaçları için devre dışı kalır ve üç şeyden biri gerçekleşene kadar uykuda kalır: CountDown() yönteminin çağrılması nedeniyle sayı sıfıra ulaşır;

veya Başka bir iş parçacığı mevcut iş parçacığını keser; veya Belirtilen bekleme süresi geçer. Sayı sıfıra ulaşırsa, yöntem true değeriyle döner. Geçerli iş parçacığı ise: kesinti durumu bu yönteme girişte ayarlanmıştır; veya Beklerken kesintiye uğrar, sonra InterruptedException atılır ve mevcut iş parçacığının kesintiye uğramış durumu temizlenir. Belirtilen bekleme süresi geçerse, false değeri döndürülür. Süre sıfırdan küçük veya sıfıra eşitse, yöntem hiç beklemeyecektir.

Soru 9:Senkron ve asenkron nedir? Aralarındaki farklar nelerdir?

Cevap9:

* Tipik olarak bir uygulamayı oluşturan iki tür programlama modeli şunlardır: senkronize ve asenkron.
* Eşzamanlı yürütme, bir programdaki ilk görevin bir sonraki göreve geçmeden önce işlemeyi bitirmesi gerektiği anlamına gelirken, eşzamansız yürütme, ikinci bir görevin önceki bir görevin bitmesini beklemeden paralel olarak yürütülmeye başlayabileceği anlamına gelir.Yani senkron aynı anda sadece bir iş yapabilme yeteneğidir.Asenkron aynı anda birden fazla iş yapabilme yeteneğidir.



*

Soru10:Paralel ve seri programlama arasındaki fark nedir?

Cevap 10:

Bilgisayar mimarisinde seri ve paralel işleme arasındaki temel fark, seri **işlemenin aynı anda tek bir görevi gerçekleştirmesi, paralel işlemenin ise aynı anda birden çok görevi gerçekleştirmesidir.**

* Bilgisayar mimarisi, bir bilgisayar sisteminin işlevselliğini, organizasyonunu ve uygulamasını tanımlar. Bilgisayar sisteminin nasıl tasarlandığını ve uyumlu olduğu teknolojileri açıklar. İşlemci , bilgisayar sistemindeki en önemli bileşenlerden biridir. Talimatları yürütür ve kendisine atanan görevleri tamamlar.
* Seri işlemede, işlemci her seferinde bir görevi tamamlar. Bunu tamamladıktan sonra, diğer görevleri sırayla yürütür. Bir işletim sistemi birçok programı yürütür ve her birinin birden fazla görevi vardır. İşlemcinin tüm bu görevleri tamamlaması gerekir, ancak her seferinde bir görevi tamamlar. Diğer görevler, işlemci geçerli görevi tamamlayana kadar kuyrukta bekler.
* Güzel bir örnek;
* Birden çok kuyruğu ve yalnızca bir kasiyeri olan bir süpermarket varsayalım. Kasiyer, bir müşterinin ürünlerini faturalamayı bitirir ve ardından başka bir müşteriye geçer. Faturalandırmayı arka arkaya gerçekleştirir.
* Paralel işlemede birden çok işlemci vardır. Her işlemci, kendisine atanan görevleri eş zamanlı olarak yürütür. İşlemciler veri yolunu birbirleriyle iletişim kurmak ve ana bellleğe erişmek

için kullanırlar . Her işlemci yerel verileri üzerinde çalışır. İşlemciler birbirinden bağımsız çalıştığı için bir işlemcideki arıza diğer işlemcinin işlevselliğini etkilemez. Bu nedenle paralel işleme, verimi artırmanın yanı sıra güvenilirliği de artırır. Çoğu modern bilgisayar, performansı artırmak için paralel işlemeyi destekler.

* Bir süpermarkette birden fazla sıra vardır ve her sıra için bir kasiyer vardır. Her kasiyer kendi kuyruğundaki müşterilerin ürünlerini faturalandırır.

* **Tanım**

* Seri işleme, bir seferde bir görevin tamamlandığı ve tüm görevlerin işlemci tarafından sırayla yürütüldüğü bir işlem türüdür. Paralel işleme, aynı anda birden çok görevin farklı işlemciler tarafından tamamlandığı bir işleme türüdür. Dolayısıyla, Seri ve Paralel İşleme arasındaki temel fark budur.

* **işlemci sayısı**

* Seri ve paralel işleme arasındaki en büyük fark, seri işlemede tek bir işlemci olması, ancak paralel işlemede birden çok işlemci olmasıdır.

* **Verim**

* Bu nedenle, paralel işlemenin performansı seri işlemeye göre daha yüksektir.

* **İş yoğunluğu**

* Seri işlemede işlemcinin iş yükü daha fazladır. Ancak paralel işlemede işlemci başına düşen iş yükü daha düşüktür. Dolayısıyla bu, seri ve paralel işleme arasındaki önemli bir farktır.

* **Veri aktarımı**
* Ayrıca, seri işlemede veri aktarımları bit bit biçimindedir. Ancak paralel işlemede veri aktarımları bayt biçimindedir (8 bit).
* **Gerekli zaman**
* Alınan süre aynı zamanda seri ve paralel işleme arasındaki bir farktır. Yani; seri işleme, bir görevi tamamlamak için paralel işlemeden daha fazla zaman gerektirir.
* **Maliyet**
* Ayrıca paralel işleme, birden fazla işlemci kullandığı için seri işlemeye göre daha maliyetlidir.

Soru 11:Javada thread kavramlarından run(),start(),wait(),notify(),notifyAll(),priority(), destroy(),getName() nedir?

Cevap11:

* run() metodu oluşturacağımız thread çalıştığı zaman çağrılacak olan metoddur ve bu metodun uygulama içindeki diğer metodlardan hiçbir farkı yoktur. İş parçacığı sınıfının run **()** yöntemi, iş parçacığı ayrı bir Runnable nesnesi kullanılarak oluşturulduysa çağrılır, aksi takdirde bu yöntem hiçbir şey yapmaz ve geri döner. run() yöntemi çağrıldığında, run() yönteminde belirtilen kod yürütülür. run() yöntemini birden çok kez çağırabilirsiniz.
* run() yöntemi, start() yöntemi kullanılarak veya run() yönteminin kendisi çağrılarak çağrılabilir.
* Thread sınıfının start **()** yöntemi, thread'in yürütülmesini başlatmak için kullanılır.

* Wait **() yöntemi,** Java'daki en üst sınıf olan Object sınıfında tanımlanır . Bu yöntem, çağıran **iş parçacığına** (Geçerli iş parçacığı) kilitten vazgeçmesini ve başka bir iş parçacığı aynı monitöre girip notify() veya notifyAll() öğesini çağırana kadar uyumaya gitmesini söyler.
* Notify()→İlgili threadin uyanmasını istiyoruz.
* NotifyAll()→Bütün threadleri ayağa kaldırır.
* Priority (Öncelik):Threadlerin bir özelliği olan priority (öncelik) 1den 10a kadar olan sayıdır.10 en yüksek öncelik, 1 en düşük önceliktir.Çoğu Virtual Machine (Sanal Makine) öncelikli thread işini bitirmedikçe veya bloke olmadıkça daha az öncelikli bir threade şans vermiyor.
* public final int getPriority():Threadin önceliğini geri çevirir.
* public final int setPriority(int yeniÖncelik):Threadin çalışma önceliğini değiştirir.
* Thread sınıfının destroy **()** yöntemi, thread grubunu ve tüm alt gruplarını yok etmek için kullanılır. Konu grubu boş olmalıdır, bu, konu grubundaki tüm konuların o zamandan beri durduğunu gösterir.
* Thread sınıfının getName **()** metodu thread ismini döndürmek için kullanılır.

Soru 12:IO işlemlerindenFileWriter, FileReader, BufferedWriter, BufferedReader nedir?

Cevap12:

* Java FileWriter sınıfı karakter yönelimli verileri bir dosyaya yazmak için kullanılır . Javada dosya işleme için kullanılan karakter yönelimli sınıftır .

* Dosyadan veri okumak için Java FileReader sınıfı kullanılır.
* BufferedWriter sınıfı ise dosyaya bir kayıt yazarken bize ayarlanabilir bir tampon bellek (Buffer) sunacaktır. Bu işlem olmadan yazma işlemi yapılması karakterlerin anında byte'a dönüştürülüp dosyaya yazılmasına yol açacak ve programın verimsiz çalışmasına yol açacaktır
* **write** metodu yazma işlemini başlatır. Dosyaya yazma işlemimiz tamamlandıktan sonra**close** komutuyla dosya ile olan işlemi tamamlarız ve diskte metin dosyamız oluşur.
* **BufferedReader**, bir girdi akışındaki metni (bir dosya gibi) karakterleri, dizileri veya satırları sorunsuz bir şekilde okuyan karakterleri tamponlayarak okuyan Java sınıfıdır.
* Genel olarak, bir Reader'dan yapılan her okuma talebi, altta yatan karakterden veya bayt akışından gelen ilgili bir okuma talebine neden olur.