Programlama Dillerinin Prensipleri

Lab Notları – 12

Java'da Thread Mantığı

```
package arayuz;

/**
 *
 * @author M.Fatih
 */
public class KarakterIslem implements Runnable {
   private char yazilanKarakter;
   private int kacKere;

   public KarakterIslem(char c,int x){
      yazilanKarakter=c;
      kacKere=x;
   }

   @Override
   public void run() {
     for(int i=0;i<kacKere;i++)
     {
        System.out.print(yazilanKarakter);
     }
     System.out.println();
   }

}
```

```
package arayuz;

/**
 *
 * @author M.Fatih
 */
public class RakamIslem implements Runnable{
   private int sonSayi;

   public RakamIslem(int sayi){
      sonSayi=sayi;
   }

   @Override
   public void run() {
      for(int i=1;i<=sonSayi;i++){
         System.out.print(" "+i);
      }
      System.out.println();
   }
```

```
}
public static void main(String[] args) {
      // TODO code application logic here

      // İşlemler tanımlanıyor
      Runnable aYaz = new KarakterIslem('a',100);
      Runnable bYaz = new KarakterIslem('b',100);
      Runnable Yaz100 = new RakamIslem(100);

      // Thread oluşturuluyor
      Thread thread1 = new Thread(aYaz);
      Thread thread2 = new Thread(bYaz);
      Thread thread3 = new Thread(Yaz100);

      // Threadler başlatılıyor
      thread1.start();
      thread2.start();
      thread3.start();
   }
```

Thread Sınıfı yield Metodu

```
@Override
   public void run() {
      for(int i=1;i<=sonSayi;i++){
         System.out.print(" "+i);
         Thread.yield();
      }
      System.out.println();
   }
```

Thread Sınıfı sleep Metodu:

```
@Override
   public void run() {
      try{

         for(int i=1;i<=sonSayi;i++){
            System.out.print(" "+i);
            if(i >= 50)Thread.sleep(1); //1 milisaniye uyu
         }
         System.out.println();
      }
      catch(InterruptedException ex){

      }
   }
```

Thread Havuzu

```
public static void main(String[] args) {

      // Havuzda 3 adet thread oluşturuluyor
      ExecutorService havuz = Executors.newFixedThreadPool(3);

      // işlemler havuza gönderiliyor
      havuz.execute(new KarakterIslem('a',100));
      havuz.execute(new KarakterIslem('b',100));
      havuz.execute(new RakamIslem(100));

      // Havuzu kapat
      havuz.shutdown();
   }
```

Thread Senkronizasyonu

```
public class Hesap {
   private int toplamPara = 0;
   public int ToplamPara(){
         return toplamPara;
   }
    public void Arttir(int miktar){
         int yeniToplam = toplamPara + miktar;

         try{
            // Veri çöküşünü daha iyi görebilmek için bekleme verildi.
            Thread.sleep(1);
         }
         catch(InterruptedException ex){
         }
         toplamPara = yeniToplam;
   }
}
```

```
public class ParaEkle implements Runnable{
   private Hesap hesap;
   public ParaEkle(Hesap hesap){
      this.hesap = hesap;
   }
   @Override
   public void run() {
      hesap.Arttir(1);
   }
}
```

```
import java.util.concurrent.*;

public class Deneme {
   public static void main(String[] args) {
      Hesap hesap = new Hesap();
      ExecutorService havuz = Executors.newFixedThreadPool(3);
```

```
        // İşlemleri oluştur
        for(int i=0;i<100;i++){
            havuz.execute(new ParaEkle(hesap));
        }
        havuz.shutdown();
        // Bütün işlemler bitene kadar bekle
        while(!havuz.isTerminated()){

        }
        System.out.println("Toplam Miktar:"+hesap.ToplamPara());

    }
}
```

```
public class Hesap {
    private int toplamPara = 0;
    private final Lock bolge = new ReentrantLock();
    public int ToplamPara(){
            return toplamPara;
    }
     public void Arttir(int miktar){
            bolge.lock();  //Kritik bölge başlangıç
            int yeniToplam = toplamPara + miktar;

            try{
               // Veri çöküşünü daha iyi görebilmek için bekleme verildi.
               Thread.sleep(1);
            }
            catch(InterruptedException ex){
            }
            toplamPara = yeniToplam;
            bolge.unlock(); //Kritik bölgeyi kapat.
    }
}
```

Basit Matris Çarpımının Thread ile Gerçekleştirimi

```
public class Matris {
    public int dizi[][];

    public Matris(Random rnd,int boyut){
        dizi = new int[boyut][boyut];

        for(int i=0;i<boyut;i++){
            for(int j=0;j<boyut;j++){
                dizi[i][j] = rnd.nextInt(10);
            }
        }
    }
    public Matris(int boyut){
        dizi = new int[boyut][boyut];
```

```
    }

    @Override
    public String toString() {
        String ekran="";
        for(int satir = 0 ; satir < dizi.length; satir++)
        {
            for (int sutun = 0 ; sutun < dizi.length; sutun++ )
            {
              ekran += "\t" + dizi[satir][sutun];
            }
            ekran += "\n";
         }
        return ekran;
    }

}
```

```
public class Carpma implements Runnable {
    private int satir;
    private int sutun;
    private Matris A;
    private Matris B;
    private Matris Sonuc;

    public Carpma(int satir, int sutun, Matris A, Matris B, Matris sonuc)    {
        this.satir = satir;
        this.sutun = sutun;
        this.A = A;
        this.B = B;
        this.Sonuc = sonuc;
    }

    @Override
    public void run()    {
        for(int i = 0; i < B.dizi.length; i++)       {
          Sonuc.dizi[satir][sutun] += A.dizi[satir][i] * B.dizi[i][sutun];
        }
    }
}
```

```
public class JavaApplication17 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int boyut=3;
        Random rand = new Random();

        Matris sonuc = new Matris(boyut);

        Matris A = new Matris(rand, boyut);
```

```
        Matris B = new Matris(rand, boyut);


        ExecutorService havuz = Executors.newFixedThreadPool(8);
        long baslangic = System.nanoTime(); //hesaplama başlıyor

        for(int satir = 0 ; satir < boyut; satir++)
        {
          for (int sutun = 0 ; sutun < boyut; sutun++ )         {
              havuz.execute(new Carpma(satir, sutun, A, B, sonuc));
          }
        }
       havuz.shutdown();

       while(!havuz.isTerminated()){  }

       long bitis = System.nanoTime();//hesaplama bitiyor
       double sure = (bitis-baslangic)/1000000.0;

       // A Matrisi yazdırılıyor
       System.out.println(" A Matrix : ");
       System.out.println(A);
        // B Matrisi yazdırılıyor
       System.out.println(" B Matrix : ");
       System.out.println(B);
         // Sonuç Matrisi yazdırılıyor
       System.out.println(" Sonuç : ");
       System.out.println(sonuc);

       System.out.println("\nHesaplanma Süresi " + String.format("%.2f", sure) + " milisaniye.");

    }

}
```

Hazırlayan
Dr. Öğr. Üyesi M. Fatih ADAK