GİRESUN ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
VERİ YAPILARI DERSİ VİZE SINAVI

Adı – Soyadı – Numarası:

Soru 1: Aşağıdaki iki dizinin farkı nedir?

```
int[] sayilar1 = new int[5];
```

Bu ifade 5 elemanlı boş bir dizi oluşturur. Java'daki ilkel (primitive) türler için, dizinin elemanları varsayılan değerlerle doldurulur. int türü için varsayılan değer 0'dır. Yani bu dizinin içeriği: [0, 0, 0, 0, 0]

```
int[] sayilar2 = {1, 2, 3, 4, 5};
```

Bu ifade ise doğrudan değerlerle başlatılmış bir dizi oluşturur. Yani elemanları belirlenmiştir: [1, 2, 3, 4, 5]

Soru 2: Bağlı liste (LinkedList) yapısında elemanlara erişim dizilere göre neden daha yavaştır?

Bağlı listede elemanlara erişmek için başlangıçtan itibaren düğümler tek tek takip edilir. Her eleman (düğüm) bir değer (data) ve sonraki düğümün adresini (next) tutar. Örneğin, 4. elemana ulaşmak için: Baş (head) düğümden başlanır, → 2. → 3. → 4. düğüme kadar bağlantılar (pointer'lar) izlenir. Bu işlem O(n) zaman alır.

Dizi (Array) ise farklıdır: Diziler bellekte ardışık (contiguous) alanlarda tutulur. Bu yüzden herhangi bir elemana indeks (index) ile doğrudan erişilebilir. sayilar[4]; // doğrudan 5. elemana erişim → O(1) Bu erişim sabit zamanlı (O(1)) olur.

Soru 3: Yığın (Stack) yapısı "geri alma (undo)" işlemleri için uygun mudur? Açıklayınız.

Yığın yapısı "Son Giren İlk Çıkar (LIFO – Last In, First Out)" mantığıyla çalışır. Bu, son yapılan işlemin ilk geri alınacağı anlamına gelir. Örnek: Metin düzenleyicide "Undo" işlemi. Bir kelime işlemci düşünelim: "A" harfini yazdınız "B" harfini yazdınız "C" harfini yazdınız Bu işlemler stack içine şu sırayla eklenir: [A, B, C] Bir "Undo" yaptığınızda, en son yapılan işlem ("C" ekleme işlemi) stack'ten çıkarılır (pop) ve geri alınır.

Soru 4: Yığın (Stack) ve Kuyruk (Queue) veri yapıları arasındaki farklar nelerdir?

Yığın (Stack) ve Kuyruk (Queue), her ikisi de verileri belirli bir sırayla saklayan doğrusal veri yapılarıdır; ancak çalışma mantıkları birbirinden oldukça farklıdır. Yığın, "Son Giren İlk Çıkar" (LIFO – Last In, First Out) prensibiyle çalışır. Yani en son eklenen eleman, ilk çıkarılan elemandır. Bu nedenle, veriler yalnızca bir uçtan (yığının üst kısmından) eklenip çıkarılır. Yığın yapısı; geri alma (undo) işlemleri, fonksiyon çağrıları (call stack) ve parantez dengeleme gibi durumlarda sıkça kullanılır.

Buna karşılık, Kuyruk (Queue) yapısı "İlk Giren İlk Çıkar" (FIFO – First In, First Out) prensibiyle çalışır. Burada elemanlar kuyruğun arka ucundan eklenir ve ön ucundan çıkarılır. Yani, önce eklenen eleman önce işlenir. Kuyruk yapısı; yazıcı sırası (print queue), görev planlama (task scheduling) ve grafiklerde genişlik öncelikli arama (BFS) gibi durumlarda kullanılır.

Özetle, yığın verileri ters sırayla işlerken, kuyruk verileri eklenme sırasına göre işler. Bu nedenle, uygulama gereksinimine göre hangi veri yapısının kullanılacağı, işlemlerin sırasının önemli olup olmamasına bağlıdır.

Soru 5: Aşağıdaki kod parçasında boş yerleri doldurunuz.

`class Dugum {` `  int veri;`	`class YiginYapisi {` `  int[] dizi;`

Başarılar.. Dr. Öğr. Üyesi Sercan KÜLCÜ 7.11.2025

```
    Dugum sonraki;
    Dugum(int veri) {
        this.veri = veri;
        this.sonraki = null;
    }
}
class BagliListe {
    Dugum bas;
    void basaEkle(int veri) {
        Dugum yeni = new Dugum(veri);
        yeni.sonraki = bas;
        bas = yeni;
    }
    void bastanSil() {
        if (bas != null)
            bas = bas.sonraki;
    }
}
```

```
    int tepe;
    void push(int veri) {
        if (tepe == dizi.length - 1) {
            System.out.println("dolu!");
            return;
        }
        dizi[++tepe] = veri;
    }
    int pop() {
        if (tepe == -1) {
            System.out.println("bos!");
            return -1;
        }
        return dizi[tepe--];
    }
    int top() {
        if (tepe == -1)
            return -1;
        return dizi[tepe];
    }

}
```

Başarılar.. Dr. Öğr. Üyesi Sercan KÜLCÜ 7.11.2025