*SİNEM GÜNDÜZALP*

*20200805004*

***1)***

*UML iş sistemlerinin modellenmesinde ortaya çıkmış bir dildir. İş sistemlerini, bir süreci veya herhangi bir işi grafikler ile açıklamak isteyenler tarafından kullanılır. Standartlaşmış bir yapı olduğundan dolayı, dili bilenler tarafından okunur ve aynı şekilde yorumlanır. Takım çalışmasına birebirdir. Yazılımlardaki hataları azaltmaya yarar ve sistem veya yazılım başta belirlendiği için tekrar tekrar kod yazmanın önüne geçilmiş olunur. En yaygın kullanılan UML Class diyagramlarıdır. Bu Class diyagramları sistemindeki sınıfların yapısını anlatmak için kullanılır. Nesne Tabanlı Programlama temel alınarak tasarlanmıştır. Bu diyagramların işlevleri şöyledir;*

- ***Sınıfları Gösterme****: Sistemin içinde bulunan sınıfları ve bunların özelliklerini gösterir. Sınıf adı, değişkenler ve metotlar vb bileşenlerle birlikte gösterilir.*
- ***İlişkileri Tanımlama****: Sınıflar arasındaki ilişkileri gösterir. Örneğin, inheritance, birleştirme, ilişki gibi ilişki türleri bu diyagramda gösterilir. Bu ilişkiler, sınıflar arasındaki bağıntıları, kalıtımı, iç içe geçmeyi veya bağımlılığı ifade eder.*
- ***Sınıfın Yapısını ve Davranışını Tanımlama****: Sınıfların içerdiği değişkenler, metotlar ve bu elemanlar arasındaki ilişkileri net bir şekilde gösterir. Bu, sınıfların yapısını ve davranışlarını anlamayı sağlar.*
- ***Kalıtımı ve Polimorfizmi Temsil Etme****: Kalıtım yoluyla bir sınıfın diğer bir sınıftan nasıl türediğini gösterir. Ayrıca, polimorfizm gibi nesne tabanlı programlama konseptlerini temsil edebilir.*
- ***Nesne Yönelimli Analizi ve Tasarımı Destekleme****: Sınıf diyagramları, nesne yönelimli analiz ve tasarım süreçlerinde temel bir araç olarak kullanılır.*

***2)***

*Bu 4 veri tipi Javada kullanılır farklı amaç için kullanılıp farklı performanslar gösterirler.*
***ArrayList****, dinamik bir dizidir ve elemanlar arka planda bir dizide saklanır. Elemanlara indeksle erişilebiliriz. Sıklıkla elemanların sıralı olarak saklanması ve erişilmesi gereken durumlarda kullanılır. Eleman ekleme ve erişim hızlıdır.*
***LinkedList****, çift yönlü bağlı bir listedir. Her bir eleman, bir sonraki ve bir önceki elemanın referanslarını içerir. Bu yapısı sayesinde elemanlar rastgele yerleştirilebilir ve ekleme/silme işlemleri hızlıdır. Özellikle elemanların sıkça eklendiği veya silindiği durumlarda tercih edilir. Ara elemanlara erişim maliyeti daha yüksek olabilir çünkü indis kullanılarak erişim yapılamaz.*
***HashMap****, anahtar-değer çiftlerini saklayan bir koleksiyondur. Anahtarlar benzersiz olmalıdır ve bu anahtarlarla ilişkilendirilmiş değerler bulunur. Anahtar-değer eşlemesi gerektiren durumlarda sıklıkla kullanılır. Verilerin hızlı erişimi, performanslı arama işlemleri için uygun bir veri yapısıdır.*
***HashSet****, benzersiz elemanların kümesini tutar. İçinde aynı eleman birden fazla kez bulunamaz. Elemanların sırası garanti edilmez. Özellikle veri kümesinde yinelenen elemanların olmaması gerektiği durumlarda kullanılır. Bu veri yapısı, benzersiz elemanlarla çalışmak istendiğinde tercih edilir.*

***3)*** *----*

***4)***

```
public class CaseConverter {
    public static String changeCase(String input) {
        if (input == null || input.isEmpty()) {
            return "";      // Eğer gelen input boş veya null olursa boş string dönecek
        }

        StringBuilder result = new StringBuilder();

        for (int i = 0; i < input.length(); i++) {
            char currentChar = input.charAt(i);

            if (Character.isUpperCase(currentChar)) {
                result.append(Character.toLowerCase(currentChar));

            } else if (Character.isLowerCase(currentChar)) {
                result.append(Character.toUpperCase(currentChar));

            } else {
                result.append(currentChar);    // Diğer karakterleri olduğu gibi bırakmak
için
            }
        }

        return result.toString();
    }

    public static void main(String[] args) {
        String originalText = "Merhaba Dünya!";
        String convertedText = changeCase(originalText);
        System.out.println("Orjinal: " + originalText);
        System.out.println("Dönüştürülmüş: " + convertedText);
    }
}
```

**5)**

```
public class StringOperations {
    public static String squeeze(String mainText, String charactersToRemove) {
        if (mainText == null || mainText.isEmpty() || charactersToRemove == null ||
charactersToRemove.isEmpty()) {
            return mainText; // Eğer ana metin ya da silinecek karakterler boş veya null
olursa ana metni olduğu gibi döndürecek
        }

        StringBuilder result = new StringBuilder();

        for (int i = 0; i < mainText.length(); i++) {
            char currentChar = mainText.charAt(i);

            if (charactersToRemove.indexOf(currentChar) == -1) {
                result.append(currentChar); // Eğer karakter silinmeyecekse, sonuç metnine
eklenecek
            }
        }

        return result.toString();
    }

    public static void main(String[] args) {
        String text = "Merhaba Dunya!";
        String charactersToRemove = "aeiou"; // Silinecek karakterler için

        String squeezedText = squeeze(text, charactersToRemove);

        System.out.println("Orjinal Metin: " + text);
        System.out.println("Sıkıştırılmış Metin: " + squeezedText);
    }
}
```