**StajHashAlma.java – StajHashAlma2.java Kullanılan Terimler ve İşlevler**

1. **import java.io.FileWriter;**: Java'da dosya yazmak için kullanılan sınıfı içeri aktarır. Bu sınıf, dosyalara karakter verileri yazmayı sağlar.
2. **import java.nio.file.Files;**: Java'da dosya okuma ve yazma işlemleri için kullanılan sınıfı içeri aktarır. Bu sınıf, dosyalar ve dosya sistemleri ile etkileşime geçmeyi sağlar.
3. **import java.nio.file.Paths;**: Dosya yollarını temsil eden ve Files sınıfı ile birlikte kullanılan sınıfı içeri aktarır.
4. **import java.security.MessageDigest;**: Java'da kriptografik hash fonksiyonlarını sağlayan sınıfı içeri aktarır. Bu sınıf, mesaj özetleme algoritmalarını (örneğin, SHA-256) kullanmayı sağlar.
5. **public class StajHashAlma {**: stajhashalma adlı bir sınıf tanımlar. Java'da sınıflar, kodun temel yapı taşlarıdır.
6. **public static void main(String[] args) {**: Java programının ana giriş noktası olan main metodunu tanımlar. Program, bu metottan başlar.
7. **try {**: Hata yakalama (exception handling) bloğunu başlatır. Bu blok içindeki kod çalıştırılır ve eğer bir hata (exception) oluşursa, bu hata catch bloğunda yakalanır.
8. **byte[] fileBytes = Files.readAllBytes(Paths.get("Albil\_staj.txt"));**:
   * Paths.get("Albil\_staj.txt"): Belirtilen dosya yolunu temsil eder.
   * Files.readAllBytes(...): Bu dosyayı byte dizisi olarak okur.
   * byte[] fileBytes: Dosyadan okunan byte'ları tutan bir byte dizisi.
9. **MessageDigest md = MessageDigest.getInstance("SHA-256");**:
   * MessageDigest.getInstance("SHA-256"): SHA-256 algoritmasını kullanmak için bir MessageDigest nesnesi oluşturur.
   * MessageDigest md: Bu nesneyi tutan değişken.
10. **byte[] hashBytes = md.digest(fileBytes);**:
    * md.digest(fileBytes): fileBytes dizisinin SHA-256 hash'ini hesaplar.
    * byte[] hashBytes: Hesaplanan hash değerini tutan byte dizisi.
11. **StringBuilder hashString = new StringBuilder();**:
    * StringBuilder: Java'da verimli şekilde değiştirilebilir stringler oluşturmak için kullanılan sınıf.
    * hashString: Hash değerinin hex (onaltılık) temsilini oluşturmak için kullanılan StringBuilder nesnesi.
12. **for (byte b : hashBytes) { hashString.append(String.format("%02x", b)); }**:
    * for (byte b : hashBytes): hashBytes dizisindeki her byte'ı iteratif olarak dolaşır.
    * String.format("%02x", b): Her byte'ı iki basamaklı onaltılık stringe çevirir.
    * hashString.append(...): Bu onaltılık stringi hashString nesnesine ekler.
13. **FileWriter writer = new FileWriter("Albil\_staj\_hash.txt");**:
    * FileWriter("Albil\_staj\_hash.txt"): Belirtilen dosyayı yazmak için bir FileWriter nesnesi oluşturur.
    * FileWriter writer: Bu nesneyi tutan değişken.
14. **writer.write(hashString.toString());**:
    * hashString.toString(): hashString içeriğini string olarak alır.
    * writer.write(...): Bu stringi dosyaya yazar.
15. **writer.close();**: Dosya yazma işlemi tamamlandıktan sonra FileWriter nesnesini kapatır.
16. **System.out.println("Hash, Albil\_staj\_hash.txt dosyasına yazıldı.");**:
    * Konsola "Hash, Albil\_staj\_hash.txt dosyasına yazıldı." mesajını yazdırır.
17. **catch (Exception e) { e.printStackTrace(); }**:
    * Eğer try bloğu içinde bir hata (exception) oluşursa, bu blok çalıştırılır.
    * e.printStackTrace(): Hatanın ayrıntılarını konsola yazdırır.

Bu kod, belirtilen dosyanın (Albil\_staj.txt) SHA-256 hash değerini hesaplar ve bu hash değerini yeni bir dosyaya (Albil\_staj\_hash.txt) yazar.

**StajSifreleme.java Kullanılan Terimler ve İşlevler**

1. **import java.io.FileOutputStream;**: Java'da dosyalara byte verisi yazmak için kullanılan sınıfı içeri aktarır.
2. **import java.nio.file.Files;**: Java'da dosya okuma ve yazma işlemleri için kullanılan sınıfı içeri aktarır. Bu sınıf, dosyalar ve dosya sistemleri ile etkileşime geçmeyi sağlar.
3. **import java.nio.file.Paths;**: Dosya yollarını temsil eden ve Files sınıfı ile birlikte kullanılan sınıfı içeri aktarır.
4. **import javax.crypto.Cipher;**: Java'da şifreleme işlemleri için kullanılan sınıfı içeri aktarır. Bu sınıf, şifreleme ve şifre çözme işlemlerini sağlar.
5. **import javax.crypto.KeyGenerator;**: Java'da kriptografik anahtarlar oluşturmak için kullanılan sınıfı içeri aktarır.
6. **import javax.crypto.SecretKey;**: Java'da simetrik şifreleme anahtarlarını temsil eden sınıfı içeri aktarır.
7. **public class stajsifreleme {**: stajsifreleme adlı bir sınıf tanımlar. Java'da sınıflar, kodun temel yapı taşlarıdır.
8. **public static void main(String[] args) {**: Java programının ana giriş noktası olan main metodunu tanımlar. Program, bu metottan başlar.
9. **try {**: Hata yakalama (exception handling) bloğunu başlatır. Bu blok içindeki kod çalıştırılır ve eğer bir hata (exception) oluşursa, bu hata catch bloğunda yakalanır.
10. **byte[] hashBytes = Files.readAllBytes(Paths.get("Albil\_staj\_hash.txt"));**:
    * Paths.get("Albil\_staj\_hash.txt"): Belirtilen dosya yolunu temsil eder.
    * Files.readAllBytes(...): Bu dosyayı byte dizisi olarak okur.
    * byte[] hashBytes: Dosyadan okunan byte'ları tutan bir byte dizisi.
11. **KeyGenerator keyGen = KeyGenerator.getInstance("AES");**:
    * KeyGenerator.getInstance("AES"): AES algoritması için bir KeyGenerator nesnesi oluşturur.
    * KeyGenerator keyGen: Bu nesneyi tutan değişken.
12. **keyGen.init(128);**: AES algoritması için 128-bit uzunluğunda bir anahtar oluşturur.
13. **SecretKey secretKey = keyGen.generateKey();**:
    * keyGen.generateKey(): AES algoritması için bir gizli anahtar (secret key) oluşturur.
    * SecretKey secretKey: Bu anahtarı tutan değişken.
14. **Cipher cipher = Cipher.getInstance("AES");**:
    * Cipher.getInstance("AES"): AES algoritması için bir Cipher nesnesi oluşturur.
    * Cipher cipher: Bu nesneyi tutan değişken.
15. **cipher.init(Cipher.ENCRYPT\_MODE, secretKey);**:
    * cipher.init(Cipher.ENCRYPT\_MODE, secretKey): Cipher nesnesini şifreleme modunda başlatır ve gizli anahtarı kullanır.
16. **byte[] encryptedHash = cipher.doFinal(hashBytes);**:
    * cipher.doFinal(hashBytes): hashBytes dizisini şifreler.
    * byte[] encryptedHash: Şifrelenmiş byte'ları tutan bir byte dizisi.
17. **FileOutputStream fos = new FileOutputStream("Albil\_staj\_hash\_encrypted.txt");**:
    * FileOutputStream("Albil\_staj\_hash\_encrypted.txt"): Belirtilen

dosyayı yazmak için bir FileOutputStream nesnesi oluşturur. - FileOutputStream fos: Bu nesneyi tutan değişken.

1. **fos.write(encryptedHash);**:
   * encryptedHash: Şifrelenmiş byte'ları dosyaya yazar.
2. **fos.close();**: Dosya yazma işlemi tamamlandıktan sonra FileOutputStream nesnesini kapatır.
3. **System.out.println("Albil\_staj\_hash.txt dosyası şifrelendi.");**:
   * Konsola "Albil\_staj\_hash.txt dosyası şifrelendi." mesajını yazdırır.
4. **try (FileOutputStream keyOut = new FileOutputStream("secretKey.key")) {**:
   * FileOutputStream("secretKey.key"): Şifreleme anahtarını saklamak için belirtilen dosyayı yazmak üzere bir FileOutputStream nesnesi oluşturur.
   * FileOutputStream keyOut: Bu nesneyi tutan değişken.
5. **keyOut.write(secretKey.getEncoded());**:
   * secretKey.getEncoded(): Gizli anahtarı byte dizisi olarak alır.
   * keyOut.write(...): Bu byte dizisini dosyaya yazar.
6. **} catch (Exception e) { e.printStackTrace(); }**:
   * Eğer try bloğu içinde bir hata (exception) oluşursa, bu blok çalıştırılır.
   * e.printStackTrace(): Hatanın ayrıntılarını konsola yazdırır.

Bu kod, belirtilen hash dosyasını (Albil\_staj\_hash.txt) AES algoritması ile şifreler ve şifrelenmiş veriyi (Albil\_staj\_hash\_encrypted.txt) ve şifreleme anahtarını (secretKey.key) yeni dosyalara yazar. Anahtar, daha sonra şifre çözme işlemi için kullanılır.

**StajSifreCozme.java Kullanılan Terimler ve İşlevler**

1. **import java.nio.file.Files;: Java'da dosya okuma ve yazma işlemleri için kullanılan sınıfı içeri aktarır. Bu sınıf, dosyalar ve dosya sistemleri ile etkileşime geçmeyi sağlar.**
2. **import java.nio.file.Paths;: Dosya yollarını temsil eden ve Files sınıfı ile birlikte kullanılan sınıfı içeri aktarır.**
3. **import javax.crypto.Cipher;: Java'da şifreleme işlemleri için kullanılan sınıfı içeri aktarır. Bu sınıf, şifreleme ve şifre çözme işlemlerini sağlar.**
4. **import javax.crypto.SecretKey;: Java'da simetrik şifreleme anahtarlarını temsil eden sınıfı içeri aktarır.**
5. **import javax.crypto.spec.SecretKeySpec;: Byte dizilerinden gizli anahtar (secret key) oluşturmaya yarayan sınıfı içeri aktarır.**
6. **public class stajsifrecozme {: stajsifrecozme adlı bir sınıf tanımlar. Java'da sınıflar, kodun temel yapı taşlarıdır.**
7. **public static void main(String[] args) {: Java programının ana giriş noktası olan main metodunu tanımlar. Program, bu metottan başlar.**
8. **try {: Hata yakalama (exception handling) bloğunu başlatır. Bu blok içindeki kod çalıştırılır ve eğer bir hata (exception) oluşursa, bu hata catch bloğunda yakalanır.**
9. **byte[] encryptedHash = Files.readAllBytes(Paths.get("Albil\_staj\_hash\_encrypted.txt"));:**
   * **Paths.get("Albil\_staj\_hash\_encrypted.txt"): Belirtilen dosya yolunu temsil eder.**
   * **Files.readAllBytes(...): Bu dosyayı byte dizisi olarak okur.**
   * **byte[] encryptedHash: Dosyadan okunan şifrelenmiş byte'ları tutan bir byte dizisi.**
10. **byte[] keyBytes = Files.readAllBytes(Paths.get("secretKey.key"));:**
    * **Paths.get("secretKey.key"): Belirtilen dosya yolunu temsil eder.**
    * **Files.readAllBytes(...): Bu dosyayı byte dizisi olarak okur.**
    * **byte[] keyBytes: Dosyadan okunan anahtar byte'larını tutan bir byte dizisi.**
11. **SecretKey secretKey = new SecretKeySpec(keyBytes, 0, keyBytes.length, "AES");:**
    * **new SecretKeySpec(keyBytes, 0, keyBytes.length, "AES"): Byte dizisinden (keyBytes) bir SecretKey nesnesi oluşturur. Bu, AES algoritması için kullanılır.**
    * **SecretKey secretKey: Bu nesneyi tutan değişken.**
12. **Cipher cipher = Cipher.getInstance("AES");:**
    * **Cipher.getInstance("AES"): AES algoritması için bir Cipher nesnesi oluşturur.**
    * **Cipher cipher: Bu nesneyi tutan değişken.**
13. **cipher.init(Cipher.DECRYPT\_MODE, secretKey);:**
    * **cipher.init(Cipher.DECRYPT\_MODE, secretKey): Cipher nesnesini şifre çözme modunda başlatır ve gizli anahtarı kullanır.**
14. **byte[] decryptedHash = cipher.doFinal(encryptedHash);:**
    * **cipher.doFinal(encryptedHash): encryptedHash dizisinin şifresini çözer.**
    * **byte[] decryptedHash: Şifresi çözülmüş byte'ları tutan bir byte dizisi.**
15. **String decryptedHashString = new String(decryptedHash);:**
    * **new String(decryptedHash): decryptedHash dizisini bir stringe çevirir.**
    * **String decryptedHashString: Şifresi çözülmüş hash'in string olarak tutulduğu değişken.**
16. **System.out.println("Şifre çözülmüş hash: " + decryptedHashString);:**
    * **Konsola "Şifre çözülmüş hash: " ve şifresi çözülmüş hash stringini yazdırır.**
17. **} catch (Exception e) { e.printStackTrace(); }:**
    * **Eğer try bloğu içinde bir hata (exception) oluşursa, bu blok çalıştırılır.**
    * **e.printStackTrace(): Hatanın ayrıntılarını konsola yazdırır.**

**Bu kod, belirtilen şifrelenmiş dosyayı (Albil\_staj\_hash\_encrypted.txt) AES algoritması ile çözer ve şifrelenmiş veriyi (hash değeri) geri elde eder. Daha sonra şifre çözülmüş hash değerini konsola yazdırır.**

**HashKarsilastirma.java Kullanılan Terimler ve İşlevler**

1. **import java.nio.file.Files;:**
   * **Java'da dosya okuma ve yazma işlemleri için kullanılan sınıfı içeri aktarır.**
2. **import java.nio.file.Paths;:**
   * **Dosya yollarını temsil eden ve Files sınıfı ile birlikte kullanılan sınıfı içeri aktarır.**
3. **import javax.crypto.Cipher;:**
   * **Java'da şifreleme ve şifre çözme işlemleri için kullanılan sınıfı içeri aktarır.**
4. **import javax.crypto.SecretKey;:**
   * **Java'da simetrik şifreleme anahtarlarını temsil eden sınıfı içeri aktarır.**
5. **import javax.crypto.spec.SecretKeySpec;:**
   * **Byte dizilerinden gizli anahtar oluşturmayı sağlayan sınıfı içeri aktarır.**
6. **public class sonuc {:**
   * **sonuc adlı bir sınıf tanımlar.**
7. **public static void main(String[] args) {:**
   * **Java programının ana giriş noktası olan main metodunu tanımlar. Program buradan başlar.**
8. **try {:**
   * **Hata yakalama bloğunu başlatır. Bu blok içindeki kod çalıştırılır ve bir hata oluşursa, catch bloğunda yakalanır.**
9. **String originalHash = new String(Files.readAllBytes(Paths.get("Albil\_staj\_hash.txt")));:**
   * **Paths.get("Albil\_staj\_hash.txt"): Hash dosyasının yolunu temsil eder.**
   * **Files.readAllBytes(...): Dosyayı byte dizisi olarak okur.**
   * **new String(...): Byte dizisini bir stringe dönüştürür.**
   * **String originalHash: İlk hash değerini tutan string.**
10. **String modifiedHash = new String(Files.readAllBytes(Paths.get("Albil\_staj\_hash\_2.txt")));:**
    * **Paths.get("Albil\_staj\_hash\_2.txt"): Hash dosyasının yolunu temsil eder.**
    * **Files.readAllBytes(...): Dosyayı byte dizisi olarak okur.**
    * **new String(...): Byte dizisini bir stringe dönüştürür.**
    * **String modifiedHash: Değiştirilmiş hash değerini tutan string.**
11. **byte[] encryptedHash = Files.readAllBytes(Paths.get("Albil\_staj\_hash\_encrypted.txt"));:**
    * **Paths.get("Albil\_staj\_hash\_encrypted.txt"): Şifrelenmiş hash dosyasının yolunu temsil eder.**
    * **Files.readAllBytes(...): Dosyayı byte dizisi olarak okur.**
    * **byte[] encryptedHash: Şifrelenmiş hash'i tutan byte dizisi.**
12. **byte[] keyBytes = Files.readAllBytes(Paths.get("secretKey.key"));:**
    * **Paths.get("secretKey.key"): Şifreleme anahtarının dosya yolunu temsil eder.**
    * **Files.readAllBytes(...): Dosyayı byte dizisi olarak okur.**
    * **byte[] keyBytes: Şifreleme anahtarını tutan byte dizisi.**
13. **SecretKey secretKey = new SecretKeySpec(keyBytes, 0, keyBytes.length, "AES");:**
    * **new SecretKeySpec(keyBytes, 0, keyBytes.length, "AES"): Byte dizisinden (keyBytes) AES algoritması için bir SecretKey nesnesi oluşturur.**
    * **SecretKey secretKey: Bu nesneyi tutan değişken.**
14. **Cipher cipher = Cipher.getInstance("AES");:**
    * **Cipher.getInstance("AES"): AES algoritması için bir Cipher nesnesi oluşturur.**
    * **Cipher cipher: Bu nesneyi tutan değişken.**
15. **cipher.init(Cipher.DECRYPT\_MODE, secretKey);:**
    * **cipher.init(Cipher.DECRYPT\_MODE, secretKey): Cipher nesnesini şifre çözme modunda başlatır ve gizli anahtarı kullanır.**
16. **byte[] decryptedHashBytes = cipher.doFinal(encryptedHash);:**
    * **cipher.doFinal(encryptedHash): Şifrelenmiş hash'in şifresini çözer.**
    * **byte[] decryptedHashBytes: Şifre çözülmüş hash byte dizisini tutar.**
17. **String decryptedHash = new String(decryptedHashBytes);:**
    * **new String(decryptedHashBytes): Şifre çözülmüş byte dizisini bir stringe dönüştürür.**
    * **String decryptedHash: Şifre çözülmüş hash değerini tutan string.**
18. **if (originalHash.equals(modifiedHash)) {:**
    * **originalHash.equals(modifiedHash): İlk hash ile değiştirilmiş hash'in eşit olup olmadığını kontrol eder.**
19. **System.out.println("Dosya değişmemiş");:**
    * **Eğer hash'ler eşitse, konsola "Dosya değişmemiş" mesajını yazdırır.**
20. **} else {:**
    * **Hash'ler eşit değilse, bu blok çalışır.**
21. **System.out.println("Dosyanız değiştirilmiştir");:**
    * **Eğer hash'ler farklıysa, konsola "Dosyanız değiştirilmiştir" mesajını yazdırır.**
22. **} catch (Exception e) { e.printStackTrace(); }:**
    * **Eğer try bloğu içinde bir hata (exception) oluşursa, bu blok çalıştırılır.**
    * **e.printStackTrace(): Hatanın ayrıntılarını konsola yazdırır.**

**Bu kod, orijinal ve değiştirilmiş hash değerlerini karşılaştırarak dosyanın değiştirilip değiştirilmediğini belirler. Ayrıca, şifrelenmiş hash değerini çözerek elde edilen hash'i de karşılaştırır.**