Assignment No: 4

Title: Implement MD5 Hashing: Calculate the MD5 message digest of a text input using Java

Name: Shubhankar jakate
PRN : 22310371
Roll No = 382019

Objective:

1. To study the concept of **Message Digests** in Information Security.
2. To understand the **MD5 hashing algorithm**.
3. To implement MD5 hashing for text input using Java.

Theory:

Introduction to Hashing

Hashing is a process of converting input data (message) into a **fixed-size string of characters**, which is typically a hexadecimal number.

- The result of hashing is called a **hash value** or **message digest**.
- Hashing is a **one-way function** → once data is hashed, it cannot be reverted to its original form.
- Hashing is widely used in **data integrity verification, password storage, and digital signatures**.

MD5 (Message Digest Algorithm 5)

- Developed by **Ronald Rivest** in **1991**.
- Produces a **128-bit hash value** (32 hexadecimal characters).
- **Deterministic** → same input will always produce the same output.
- **Fixed Length** → regardless of input size, output is always 128 bits.

Working of MD5 Algorithm

MD5 processes the input message in blocks of **512 bits** (64 bytes) and produces a **128-bit digest**.

1. **Padding:**
   - The original message is padded so that its length (in bits) is congruent to 448 modulo 512.
   - Padding always starts with a `1` bit followed by `0`s.
2. **Appending Length:**
   - The original length of the message (in bits) is appended as a 64-bit number.
3. **Initialize Buffers:**
   - Four 32-bit variables (A, B, C, D) are initialized with fixed constants.
4. **Processing in Blocks:**
   - Each 512-bit block is divided into 16 words of 32 bits.
   - Nonlinear functions (F, G, H, I) and bitwise operations are applied in **64 rounds**.
5. **Output:**
   - After processing all blocks, the final values of (A, B, C, D) are concatenated to form the **128-bit digest**.

Applications of MD5

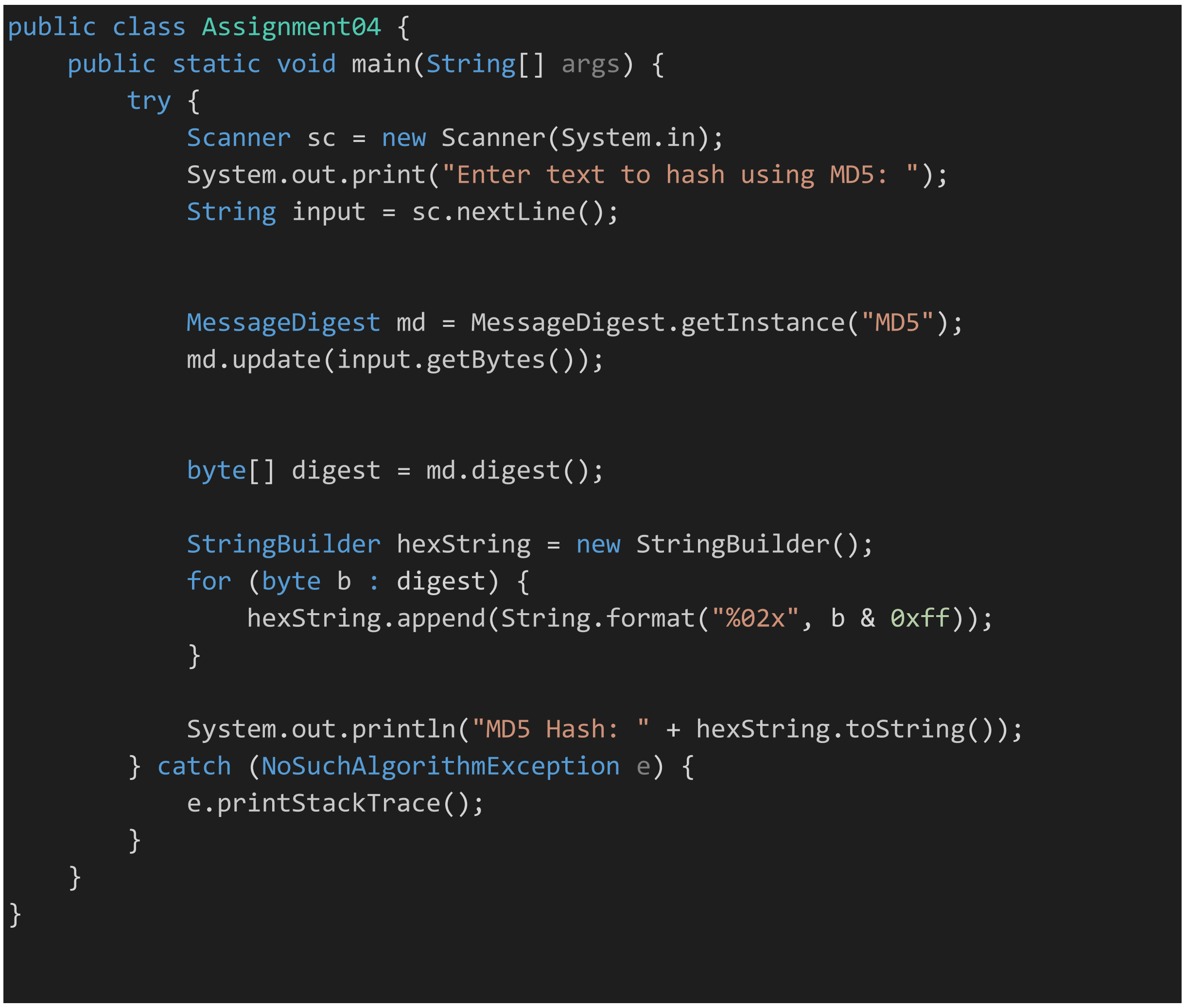
- **Password storage** (although now considered weak).
- **Data integrity check** → ensuring file not corrupted during transfer.
- **Digital signatures** (used inside certificates).
- **Checksums** → verifying data consistency.

Limitations of MD5

- **Collision attacks possible**: Different inputs can generate the same hash.
- **Not suitable for modern security** (replaced by SHA-256, SHA-3).
- **Still used** in non-critical tasks like checksums, but not for cryptographic security.

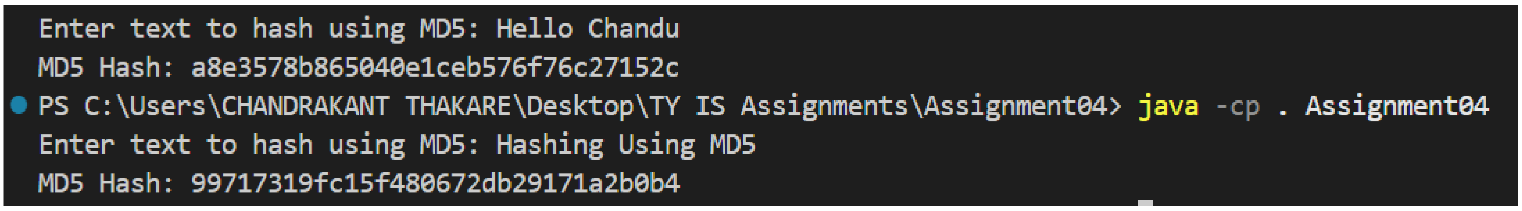
Code:

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Scanner;
```



```java
public class Assignment04 {
    public static void main(String[] args) {
        try {
            Scanner sc = new Scanner(System.in);
            System.out.print("Enter text to hash using MD5: ");
            String input = sc.nextLine();


            MessageDigest md = MessageDigest.getInstance("MD5");
            md.update(input.getBytes());


            byte[] digest = md.digest();

            StringBuilder hexString = new StringBuilder();
            for (byte b : digest) {
                hexString.append(String.format("%02x", b & 0xff));
            }

            System.out.println("MD5 Hash: " + hexString.toString());
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
    }
}
```

OUTPUT :



```
  Enter text to hash using MD5: Hello Chandu
  MD5 Hash: a8e3578b865040e1ceb576f76c27152c
● PS C:\Users\CHANDRAKANT THAKARE\Desktop\TY IS Assignments\Assignment04> java -cp . Assignment04
  Enter text to hash using MD5: Hashing Using MD5
  MD5 Hash: 99717319fc15f480672db29171a2b0b4
```