Pt1 — Verificació d'integritat mitjançant hash (Java)

1.Objectiu

Desenvolupar un programa en **Java** que comprovi si un missatge de text (o el contingut d'un fitxer) ha estat modificat: s'ha de calcular el hash del missatge i comparar-lo amb un hash proporcionat per l'usuari.

2. Requisits obligatoris

1. El programa **rebrà dos arguments** per línia de comandes:
    - `--mode` amb valor `text` o `file` (indica si el segon paràmetre és text o ruta de fitxer).
    - `--input` el text o la ruta del fitxer.
    - `--hash` el hash esperat (en hexadecimal).
      Exemple d'ús:

```
java -jar VerifyHash.jar --mode text --input "Hola món" --hash d2f0...

java -jar VerifyHash.jar --mode file --input /home/alumne/missatge.txt --hash d2f0...
```

2. Utilitzar **SHA-256** com a algorisme per defecte per calcular el hash.
3. El programa ha de:
    - Llegir correctament el text o fitxer segons el mode.
    - Calcular el hash SHA-256 del contingut.
    - Comparar el hash calculat amb el hash esperat (ignorar majúscules/minúscules en la comparació hexadecimal).

- Imprimir per consola un missatge clar:
  - Si coincideixen: `VERIFICAT: El missatge NO ha estat modificat.`
  - Si no coincideixen: `ALERTA: El missatge HA ESTAT MODIFICAT (o el hash no coincideix).`

4. Gestió d'errors:
   - Missatge d'ajuda si falten arguments o són incorrectes.
   - Missatges d'error clars si el fitxer no existeix o no es pot llegir.
5. Documentació mínima: README amb instruccions d'execució i exemples.

3. Casos de prova (exemples)

1. Mode text:
   - Input: `Hola món`
   - Hash esperat (SHA-256): `d2f0df47a3cc5f0c8bb1f823e3cf92b1397c33210c3e2431c7d3e2d41f5730b6`
   - Sortida esperada: `VERIFICAT: El missatge NO ha estat modificat.`
2. Mode file:
   - Fitxer `exemple.txt` amb contingut `Prova 123`
   - Calcular hash amb una eina externa i passar-lo; el programa ha de validar correctament.
3. Comparació negativa:
   - Hash incorrecte → Sortida: `ALERTA: El missatge HA ESTAT MODIFICAT (o el hash no coincideix).`

4. Entregables

1. Codi font Java (mòduls/paquets ben estructurats).
2. Documentació: Descripció del codi i del projecte, tests i proves utilitzades.

5. Pistes i suggeriments.

- Utilitzeu `MessageDigest.getInstance("SHA-256")` per obtenir l'objecte de digest.
- Convertiu els bytes del hash a hex amb `String.format("%02x", b)` en un bucle.
- Normalitzeu el hash esperat (per exemple, `toLowerCase()` i eliminar possibles espais).
- Eviteu fer `new String(bytes)` sense indicar l'encoding quan treballeu amb textos: utilitzeu `StandardCharsets.UTF_8`.

6. Annex.

Comandes.

MessageDigest.getInstance("SHA-256")

- **Què fa?**
  Retorna un objecte `MessageDigest` configurat per a l'algorisme de hash que indiquis (`MD5`, `SHA-1`, `SHA-256`, etc.).

Exemple:

```
MessageDigest md = MessageDigest.getInstance("SHA-256");
md.update("Hola".getBytes());
byte[] hash = md.digest(); // retorna el hash final
```

MessageDigest.isEqual(byte[] digesta, byte[] digestb)

- **Què fa?**
  Compara dos arrays de bytes de forma **segura** (temps constant).
  Això evita filtracions per temporització, ja que no s'atura al primer byte diferent.

Exemple:

```
byte[] hash1 = md.digest("Hola".getBytes());
byte[] hash2 = md.digest("Hola".getBytes());

if (MessageDigest.isEqual(hash1, hash2)) {
    System.out.println("Hashes iguals!");
}
```

- **Diferència respecte `Arrays.equals()`**
  - `Arrays.equals()` s'atura en el primer byte diferent → pot donar pistes a un atacant sobre quina part coincideix.
  - `MessageDigest.isEqual()` compara sempre tot l'array → constant-time (més segur).

Conversió Hex → Bytes

Quan tens un `String` hexadecimal com ara:

`"2ef7bde608ce5404e97d5f042f95f89f1c232871"`

realment és només una representació llegible del hash (cada 2 caràcters hex representen un byte).

Exemple en Java

```
public static byte[] hexToBytes(String hex) {
    int length = hex.length();
    byte[] bytes = new byte[length / 2];
    for (int i = 0; i < length; i += 2) {
        bytes[i / 2] = (byte) ((Character.digit(hex.charAt(i),
16) << 4)
                               + Character.digit(hex.charAt(i+1),
16));
    }
    return bytes;
}
```

Conversió Bytes → Hex (el contrari)

Això ja ho fem sovint per mostrar el hash:

```
public static String bytesToHex(byte[] bytes) {
    StringBuilder sb = new StringBuilder();
    for (byte b : bytes) {
        sb.append(String.format("%02x", b));
    }
    return sb.toString();
}
```

Passar arguments a un programa Java des de

NetBeans

1. **Obrir el projecte a NetBeans.**
   Tingues el fitxer amb el `main(String[] args)` al teu projecte.

2. **Obrir les propietats del projecte.**

   - Fes clic dret al projecte → **Properties**.

3. **Anar a "Run".**
   Al menú lateral, selecciona **Run**.

Afegir Arguments.
A l'apartat **Arguments**, pots escriure els paràmetres que vols passar.
Per exemple:

```
SHA-256 "Hola món" "2ef7bde608ce5404e97d5f042f95f89f1c232871"
```

4. Això es correspondrà a:

   - `args[0] = "SHA-256"`
   - `args[1] = "Hola món"`
   - `args[2] = "2ef7bde608ce5404e97d5f042f95f89f1c232871"`

5. **Executar el projecte.**
   Quan facis `Run`, NetBeans passarà aquests arguments automàticament.