|  |
| --- |
| ***Teme:***   * *Tokovi in datoteke* * *Besedilne datoteke, binarne datoteke* * *Hash, verifikacija datotek na osnovi hash* |

|  |
| --- |
| **Ref:**  **1. http://tutorials.jenkov.com/java-cryptography/messagedigest.html**  **2. https://www.computerhope.com/issues/ch001846.htm**  **3. https://howtodoinjava.com/java/java-security/sha-md5-file-checksum-hash/** |

**Naloga 1**

Iz danih URL:

* <https://download.freebsd.org/releases/amd64/amd64/ISO-IMAGES/13.0>
* [https://github.com/notepad-plus-plus/notepad-plus-plus/releases/tag/v8.3 /](https://github.com/notepad-plus-plus/notepad-plus-plus/releases/tag/v8.3%20/)

si lokalno prenesite datoteki : npp.8.3.checksums.sha256, CHECKSUM.SHA256-FreeBSD-13.0-RELEASE-amd64.

1. Ugotovite, ali sta datoteki običajni besedilni datoteki (bralni).
2. Spišite program, ki bo s parametrom podano ali eno ali drugo datoteko odprl kot datoteko vrstic (besedilno), vsako prebrano vrstico pretvoril v objekt tipa DDatot s privatnima elementoma/lastnostima *diggest* in i*meDatoteke* in ga zapisal v kolekcijo takih elementov (vrsto kolekcije si izberite sami). Program naj pred svojim zaključkom izpiše vse elemente kolekcije v obliki:

ime programa: (<ime iz datoteke>) hash/diggest: (<diggest iz datoteke>)

…

Kolekcija vsebuje informacijo o <N> datotekah.

**Naloga 2**

Spišite program, ki bo zajel ime datoteke, nato pa zgeneriral hash te datoteke in ga izpisal na zaslon. Uporabite binarno datoteko na netipiziran način; oz. enostavno FileInputStream in njegovo metodo read. Algoritem za določanje hash funkcije naj bo sha256, testna datoteka pa poljubna datoteka (.doc, .txt, .dll). Delujoč program boste večkrat zagnali na isti datoteki in med seboj preverili generirane hash-e, ki morajo na vsaki iteraciji biti na enaki vsebini enaki!

Opomba: uporabite dane reference, program dejansko imate v zadnji ! Program uporablja branje blokov; je hitreje kot branje posameznega zloga. Hash se računa parcialno, ker je metoda njegovega delovanja taka, da deluje z bloki podatkov znane dolžine.

**Naloga 3**

Poiščite in si prenesite enega izmed programov omenjenih v npp.8.3.checksums.sha256 in programom iz naloge 2 zgenerirajte njegov hash (sha256).

1. Vizualno ga primerjate s tistim, ki je v datoteki npp.8.3.checksums.sha256 in ugotovite ali sta enaka
2. Spišite program (oz. razširite tistega iz naloge 1), da bo zgornjo primerjavo izvedel avtomatično: če dodate programu drugi parameter, ki predstavlja hash, program avtomatično ugotovi, ali podan hash pripada kateremu izmed programov v prebranem spisku in ta program tudi izpiše.

**Naloga 4**

Izvedite integracijo v naloga 1,2,3 izvedenih mehanizmov v enoten program, ki zna ali generirati ali preveriti hash (s sha256). Hash za preverjanje je pri tem lahko podan ali v datoteki ali kot niz. Glejte vir dan z drugo povezavo in interaktivni program namenjen izvajanju v okviru windows OS.

**Naloga 5**

1. Kako težko je algoritem za hash-iranje zamenjati z npr. sha512 ? Ali je postopek izvedbe na isti datoteki s sha512 daljši ? Če je, zakaj mislite, da se dalj izvaja?
2. Kaj dejansko garantirate s podpisovanjem s hash?
3. Kaj je razlog, da 'podpis' ni vgrajen v samo datoteko?