Advanced Data Structures Through Java

DOCUCHAIN: VERSION CONTROL SYSTEM

Name: M. Prashanth
Roll Number: 23b81a6297
Department: CSC-B

1. Introduction

DocuChain is a lightweight, document version control system developed using Java. It uses a combination of

Linked Lists and **HashMaps** to efficiently track changes across document versions. Inspired by systems like Git, DocuChain offers basic versioning, rollback, and tracking features without the complexity of distributed control.

---

2. Theory

The core data structures used are:

- **Linked List**:
  - Each document version is a node.
  - Nodes store metadata (version number, timestamp) and the actual content snapshot.
- **HashMap**:
  - Quick lookup for versions by version number.
  - Key = Version ID, Value = Node (Linked List).
- **Hashing**:
  - Ensures integrity by hashing document contents for verification.

Working Principle

1. **Initialization**:
   - A document is created with its first version (Version 1.0).

2. **Versioning**:
    - Any edit/addition creates a new node (Version 1.1, 1.2, etc.).
3. **Rollback**:
    - Retrieve a previous version using its version number.
4. **Hash Validation**:
    - Each version is validated using its hash to prevent corruption.

---

3. Sample Code (Java Implementation)

java

CopyEdit

```
import java.util.*;


class DocumentVersion {
   String content;
   String timestamp;
   String versionId;
   int hash;
   DocumentVersion next;
```

```
    public DocumentVersion(String content, String
versionId) {
        this.content = content;
        this.versionId = versionId;
        this.timestamp = new Date().toString();
        this.hash = content.hashCode();
    }
}

class DocuChain {
    DocumentVersion head;
    HashMap<String, DocumentVersion> versionMap = new
HashMap<>();

    public void addVersion(String content) {
        String versionId = "v" + (versionMap.size() + 1);
        DocumentVersion newVersion = new
DocumentVersion(content, versionId);
        if (head == null) {
            head = newVersion;
```

```
        } else {
            DocumentVersion temp = head;
            while (temp.next != null) {
                temp = temp.next;
            }
            temp.next = newVersion;
        }
        versionMap.put(versionId, newVersion);
        System.out.println("New version added: " + versionId);
    }

    public void viewVersion(String versionId) {
        DocumentVersion version =
versionMap.get(versionId);
        if (version != null) {
            System.out.println("Version: " + version.versionId);
            System.out.println("Timestamp: " +
version.timestamp);
            System.out.println("Content: " + version.content);
        } else {
            System.out.println("Version not found.");
```

```
    }
  }

  public void viewAllVersions() {
    DocumentVersion temp = head;
    while (temp != null) {
      System.out.println(temp.versionId + " => " +
temp.timestamp);
      temp = temp.next;
    }
  }
}
```

---

4. Input and Output

Input:

- Create a new document.
- Edit the document and create multiple versions.
- Request to view specific version(s).

Output:

- List of all versions with timestamps.
- Specific content of a version upon request.

5. Strengths

- Simple and fast document versioning.
- Quick rollback to older versions.
- Ensures data integrity with hashing.
- Lightweight compared to full version control systems.

---

6. Limitations

- Not distributed (works only on a single system).
- No merging or branching like in Git.
- Limited scalability for very large files.

---

7. Applications

- Small-scale document tracking systems.
- Academic project management tools.
- Backup systems for text files and small documents.
- Offline versioning for personal notes or logs.

8.Conclusion

DocuChain demonstrates how advanced data structures like **Linked Lists** and **HashMaps** can be used to create powerful version control mechanisms. It is a simplified, yet practical model, of real-world systems like Git, adapted for personal and academic projects.