Git hash-object Implementation

This file implements Git's `hash-object` command in Node.js. This command creates a Git blob object from a file and optionally stores it in the Git repository. Let me break it down in detail:

Purpose

The `hash-object` command takes a file, creates a Git blob object from it, calculates its SHA-1 hash, and optionally writes it to the Git object database.

Class Structure

```
class HashObjectCommand {
    constructor(flag, filepath) {
        this.flag = flag;
        this.filepath = filepath;
    }
}
```

The class takes two parameters:

- `flag`: Command-line flag (typically `-w` to write the object to the repository)
- `filepath`: Path to the file to be hashed

Step-by-Step Execution

1. File Validation

```
const filepath = path.resolve(this.filepath);

if(!fs.existsSync(filepath)){
    throw new Error(`could not open ${this.filepath} for reading: No such file or directory`)
};
```

- Converts the file path to an absolute path
- Checks if the file exists, throwing an error if not found

2. Reading File Contents

```
const fileContents = fs.readFileSync(filepath);
const fileLength = fileContents.length;
```

Reads the entire file into memory and gets its byte length.

3. Creating the Git Blob Object

```
const header = `blob ${fileLength}\0`;
const blob = Buffer.concat([Buffer.from(header), fileContents]);
```

This is the crucial part that follows Git's internal format:

- Git objects always start with a header: `{type} {size}\0`
- For files, the type is always "blob"
- The header includes the file size in bytes
- A null character (`\0`) separates the header from the content
- The final blob is the concatenation of header + file contents

Example: If you have a file with content "Hello World" (11 bytes), the blob would be:

blob 11\0Hello World

4. Calculating the SHA-1 Hash

```
const hash = crypto.createHash("sha1").update(blob).digest("hex");
```

- Creates a SHA-1 hash of the entire blob (header + content)
- This hash becomes the unique identifier for this object in Git
- The hash is returned as a 40-character hexadecimal string

5. Optional Storage (with `-w` flag)

```
if(this.flag && this.flag === '-w'){
   const folder = hash.slice(0, 2);
   const file = hash.slice(2);

   const completeFolderPath = path.join(process.cwd(), '.git', 'objects', folder);

   if(!fs.existsSync(completeFolderPath)){
      fs.mkdirSync(completeFolderPath, { recursive: true });
   }
```

```
    //compress the blob
    const compressData = zlib.deflateSync(blob)
    fs.writeFileSync(
        path.join(completeFolderPath, file), compressData
    );
}
```

When the `-w` flag is provided:

1. **Directory Structure**: Uses Git's standard object storage format
    - First 2 characters of hash become the directory name
    - Remaining 38 characters become the filename
2. **Directory Creation**: Creates the directory if it doesn't exist
3. **Compression**: Compresses the blob using zlib (Git always compresses objects)
4. **Storage**: Writes the compressed blob to the appropriate location

6. Output

```
process.stdout.write(hash);
```

Prints the SHA-1 hash to stdout, regardless of whether the object was written to disk.

Example Usage Scenarios

Without `-w` flag (just calculate hash):

```
node hash-object.js README.md
# Output: e69de29bb2d1d6434b8b29ae775ad8c2e48c5391
```

With `-w` flag (calculate and store):

```
node hash-object.js -w README.md
# Output: e69de29bb2d1d6434b8b29ae775ad8c2e48c5391
# Also creates: .git/objects/e6/9de29bb2d1d6434b8b29ae775ad8c2e48c5391
```

Git Object Storage Structure

For hash `abc123def456...`, the file is stored as:

.git/objects/ab/c123def456...

Key Points

- **Content Addressing**: The hash uniquely identifies the content
- **Deduplication**: Identical files will always have the same hash
- **Compression**: All Git objects are compressed with zlib
- **Immutability**: Once stored, objects never change (the hash would change if content changed)

This command is fundamental to Git's operation - every file you commit goes through this process to become a blob object in Git's object database.