External Project Report on
Computer Networking (CSE3034)

[A console-based Internet Download Manager (IDM) using Java]

<u>Submitted by</u>

Name : Shiv Shankar Malla	Reg. No.: 2141013058
Name : Pritish Mishra	Reg. No.: 2141013125
Name : Neha Mohanta	Reg. No.: 2141013315
Name : Maheshwar Nag	Reg. No.: 2141014002
Name : Shivangi Nayak	Reg. No.: 2141014086

B. Tech. CSE 5th Semester (Section N)

INSTITUTE OF TECHNICAL EDUCATION AND RESEARCH
(FACULTY OF ENGINEERING)
SIKSHA 'O' ANUSANDHAN (DEEMED TO BE UNIVERSITY), BHUBANESWAR, ODISHA

Declaration

We, the undersigned students of B. Tech. of **CSE** Department hereby declare that we own the full responsibility for the information, results etc. provided in this PROJECT titled **console-based Internet Download Manager (IDM) using Java** submitted to **Siksha 'O' Anusandhan (Deemed to be University), Bhubaneswar** for the partial fulfillment of the subject **Computer Networking (CSE 3034)**. We have taken care in all respect to honor the intellectual property right and have acknowledged the contribution of othersfor using them in academic purpose and further declare that in case of any violation of intellectual property right or copyright we, as the candidate(s), will be fully responsible for the same.

Shiv Shankar Malla	**Pritish Mishra**
Regd. No.: 2141013058	**Regd. No.: 2141013125**
Neha Mohanta	**Maheshwar Nag**
Regd No.: 2141013315	**Regd No.: 2141014002**

Shivangi Nayak

Regd. No.: 2141014086

DATE: 12/01/2024

PLACE: Bhubaneshwar

Abstract

The Java code presents a rudimentary console-based Internet Download Manager (IDM) capable of downloading files from a specified URL and saving them to a designated local path. Executed from the command line, the program validates input parameters, comprising the file URL and destination path . Utilizing Java's ‘ URLConnection ’ , the ‘ downloadFile ’ method facilitates the download process by reading from the input stream and writing to the output stream . However, this implementation lacks advanced features such as multi-threading, download resumption , and progress indicators. A comprehensive download manager for real-world applications would necessitate the inclusion of these features and robust error-handling mechanisms. Additionally, implement error handling mechanisms to address network issues or unexpected interruptions during the download process. Extending the program with a graphical user interface (GUI) could further improve usability. This simple console-based IDM serves as a foundational starting point, but the evolution into a fully-featured download manager requires the integration of these advanced capabilities for optimal performance and user satisfaction in handling diverse download scenarios.

Contents

1. Introduction

The project is a console-based Internet Download Manager (IDM) implemented in Java. The IDM allows users to download files from specified URLs and save them to designated local paths. The program is designed to be executed from the command line, requiring users to provide the file URL and destination path as command-line arguments. Utilizing Java’s URLConnection , the IDM’s core functionality is encapsulated in the downloadFile method, which reads from the input stream and writes to the output stream. However, this initial implementation lacks advanced features such as multi-threading, download resumption, and progress indicators. The project's potential for improvement lies in the integration of these features, along with enhanced error-handling mechanisms and possibly the development of a graphical user interface (GUI) to enhance user interaction. This console-based IDM serves as foundational starting point, aiming to evolve into a more comprehensive download manager with enhanced capabilities and user-friendly features.

2. Problem Statement

To Develop a console-based Internet Download Manager (IDM) using Java.

Learning Outcomes:
Socket programming, HTTP request management

Features:

- **Multiple File Downloads**: Simultaneously download various files.
- **Resume Interrupted Downloads**: Capability to resume downloads from the point of interruption
- (due to network issues, system shutdown, etc.).
- **Queue Management**: Prioritize downloads, allowing users to manage the order and timing of
- their downloads.
- **File Management:** Organize downloaded files by categories and store them in specified folders.

3. Methodology

1. Start

2. Input: Accept the file URL and destination path as command-line arguments.

3. Validate Input:
    i. Ensure that both the URL and destination path are provided.
    ii. Exit the program with an error message if validation fails.

4. Download File:
    a. Define a method downloadFile(url: String, destinationPath: String):
    i.Create a URL object from the provided URL.
    ii. Open a connection to the URL using ‘URLConnection’.
    iii. Create input and output streams for reading from the connection and writing to the local file.
    iv. Initialize a buffer for data transfer.

    b. Use a loop to read data from the input stream and write it to the output stream until the end of the file is reached.

    c. Close the input and output streams.

5. Main Program:
    a. Invoke the downloadFile method with the provided URL and destination path.

    b. Handle exceptions:
    i. Catch and handle IOExceptions that may occur during the download process.

    c. Print a success message if the download is completed successfully.

6. End

4.Implementation

Download.java

```
import java.io.InputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URL;

// This class downloads a file from a URL.
class Download implements Runnable {

// Max size of download buffer.
private static final int MAX_BUFFER_SIZE = 1024;

// These are the status names.
public static final String STATUSES[] = {"Downloading", "Paused", "Complete",
"Cancelled", "Error"};

// These are the status codes.
public static final int DOWNLOADING = 0;
public static final int PAUSED = 1;
public static final int COMPLETE = 2;
public static final int CANCELLED = 3;
public static final int ERROR = 4;

private URL url; // download URL
private long size; // size of download in bytes private long downloaded; // number
of bytes downloaded private int status; // current status of download
private long initTime; //inital time when download started or resumed private long
startTime; // start time for current bytes
private long readSinceStart; // number of bytes downloaded since startTime private
long elapsedTime=0; // elapsed time till now
private long prevElapsedTime=0; // time elapsed before resuming download private
long remainingTime=-1; //time remaining to finish download private float avgSpeed=0;
//average download speed in KB/s
private float speed=0; //download speed in KB/s
// Constructor for Download. public Download(URL url) {
this.url = url;
size = -1;
downloaded = 0;
status = DOWNLOADING;
// Begin the download.download();
}
// Get this download's URL.
```

```
public String getUrl() {
return url.toString();
}
// Get this download's size.
public long getSize() {
return size;
}
// Get download speed.
public float getSpeed() {
return speed;
}
// Get average speed
public float getAvgSpeed() {
return avgSpeed;
}
// Get elapsed time
public String getElapsedTime() {
return formatTime(elapsedTime/1000000000);
}
// Get remaining time
public String getRemainingTime() {
if(remainingTime<0)
return "Unknown";
else
return formatTime(remainingTime);
}
// Format time
public String formatTime(long time) { //time in seconds String s="";
s+=(String.format("%02d", time/3600))+":";
time%=3600;
s+=(String.format("%02d", time/60))+":";
time%=60;
s+=String.format("%02d", time);
return s;
}
// Get this download's progress. public float getProgress() {
return ((float) downloaded / size) * 100;
}
// Get this download's status. public int getStatus() {
return status;
}
// Pause this download. public void pause() {
prevElapsedTime=elapsedTime;
status = PAUSED;
}
// Resume this download. public void resume() {
status = DOWNLOADING;
download();
```

```
prevElapsedTime=elapsedTime;
status = CANCELLED;
}

// Mark this download as having an error.
private void error() {
prevElapsedTime=elapsedTime;
status = ERROR;
System.out.println("Sorry your file type is not valid:");
}

// Start or resume downloading.
private void download() {
Thread thread = new Thread(this);
thread.start();
}

// Get file name portion of URL.
public String getFileName(URL url) {
String fileName = url.getFile();
return fileName.substring(fileName.lastIndexOf('/') + 1);
}

// Download file.
public void run() {
RandomAccessFile file = null;
InputStream stream = null;

try {
// Open connection to URL.
HttpURLConnection connection = (HttpURLConnection) url.openConnection();

// Specify what portion of file to download.
connection.setRequestProperty("Range", "bytes=" +downloaded +"-");

// Connect to server.
connection.connect();
System.out.println("Connected to the server, Response:
"+connection.getResponseMessage());

// Make sure response code is in the 200 range.
if (connection.getResponseCode() / 100 != 2) {
error();
}

// Check for valid content length.
int contentLength = connection.getContentLength();
if (contentLength < 1) {
```

```
error();
}

/* Set the size for this download if it hasn't been already set. */
if (size == -1) {
size = contentLength;
}
// used to update speed at regular intervals
int i=0;
// Open file and seek to the end of it.
file = new RandomAccessFile(getFileName(url), "rw");
file.seek(downloaded);

stream = connection.getInputStream();
initTime = System.nanoTime();
while (status == DOWNLOADING) {
/* Size buffer according to how much of the file is left to download. */
if(i==0)
{
    startTime = System.nanoTime();
    readSinceStart = 0;
}
byte buffer[];
if (size - downloaded > MAX_BUFFER_SIZE) {
buffer = new byte[MAX_BUFFER_SIZE];
} else {
buffer = new byte[(int)(size - downloaded)];
}
// Read from server into buffer.
int read = stream.read(buffer);
if (read == -1)
break;
// Write buffer to file.
file.write(buffer, 0, read);
downloaded += read;
readSinceStart+=read;
//update speed
i++;
if(i>=50)
{   speed=(readSinceStart*976562.5f)/(System.nanoTime()-startTime);

if(speed>0) remainingTime=(long)((size-
 downloaded)/(speed*1024));
else remainingTime=-1;
elapsedTime=prevElapsedTime+(System.nanoTime()-initTime);
avgSpeed=(downloaded*976562.5f)/elapsedTime;
i=0;
}
```

```
}

/* Change status to complete if this point was reached because downloading has
finished. */
if (status == DOWNLOADING) {
    status = COMPLETE;
}
} catch (Exception e) {
System.out.println(e);
error();
} finally {
// Close file.
if (file != null) {
try {
file.close();
} catch (Exception e) {
System.out.println(e);
}
}

// Close connection to server.
if (stream != null) {
try {
stream.close();
} catch (Exception e) {
System.out.println(e);
}
}
}
}

}
```

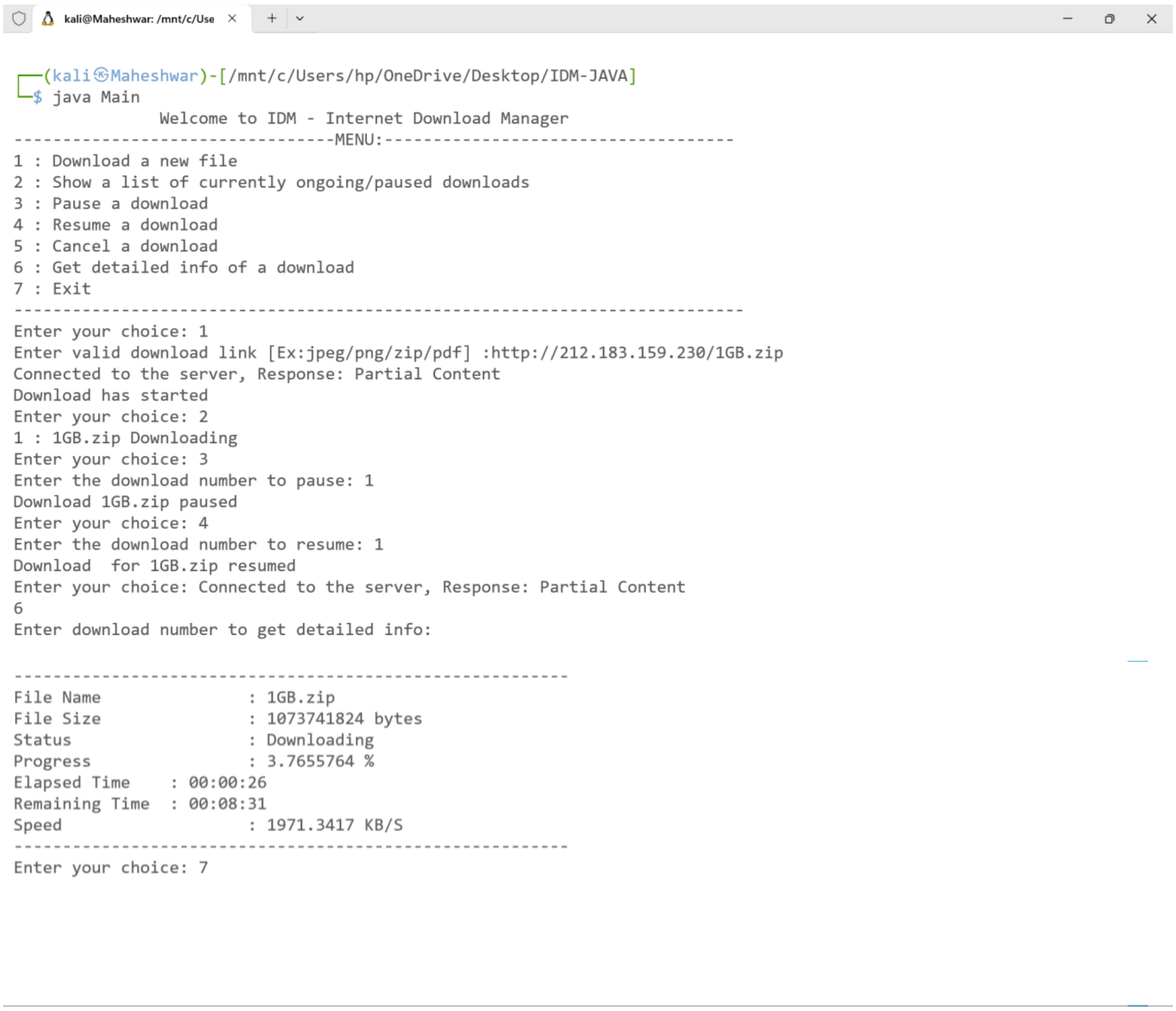
Main.java

```
import java.util.ArrayList;
import java.util.Scanner;
public class Main {
private static Scanner s;
public static void main(String[] args) throws Exception {
System.out.println("  Welcome to IDM - Internet DownloadManager");

System.out.println("----------------------------------------MENU:-------------------
---------------------------------------");
System.out.println("1 : Download a new file");
System.out.println("2 : Show a list of currently ongoing/paused downloads");
System.out.println("3 : Pause a download");
System.out.println("4 : Resume a download");
System.out.println("5 : Cancel a download");
System.out.println("6 : Get detailed info of a download");
System.out.println("7 : Exit");
System.out.println("----------------------------------------------------------------
-------------------------");

ArrayList<Download> downloads = new ArrayList<Download>();

while(true) {
s = new Scanner(System.in);
System.out.print("Enter your choice: ");
int choice = s.nextInt();
switch(choice) {
case 1:
System.out.print("Enter valid download link [Ex:jpeg/png/zip/pdf] :");
s.nextLine();
String link = s.nextLine();
Download d = new Download(new URL(link));
while(d.getProgress()<=0) {}
System.out.println("Download has started");
downloads.add(d);
break;
case 2:
if(downloads.size()==0) {
System.out.println("There are currently no ongoing downloads!");

break;
}

for(int i = 0;i<downloads.size();i++) {
Download temp = downloads.get(i);
String status;
if(temp.getStatus()==1) status = "PAUSED";
else if(temp.getStatus()==0) status = "Downloading";
else status = "N/A";
if(temp.getStatus()!=2 && temp.getStatus()!=4) {
System.out.println(i+1 +" : " +temp.getFileName(new
URL(temp.getUrl())) +" " +status);
}if(temp.getStatus()==2||temp.getStatus()==4)
downloads.remove(i);
}
break;
case 3:
System.out.print("Enter the download number to pause: ");
int num = s.nextInt();
if(downloads.get(num-1).getStatus()==1)
System.out.println("The download is already paused!");
else {
downloads.get(num-1).pause();
System.out.println("Download " +downloads.get(num-1).getFileName(new
URL(downloads.get(num-1).getUrl())) +" paused");
}
break;
case 4:
System.out.print("Enter the download number to resume: ");
num = s.nextInt();
if(downloads.get(num-1).getStatus()==0)
System.out.println("The file is already downloading!");
else {
downloads.get(num-1).resume();
System.out.println("Download for " +downloads.get(num-1).getFileName(new
URL(downloads.get(num-1).getUrl())) +" resumed");
}
break;
case 5:
System.out.print("Enter download number to cancel: ");
num = s.nextInt();
downloads.get(num-1).cancel();
System.out.println("Download for " +downloads.get(num-1).getFileName(new
URL(downloads.get(num-1).getUrl())) +" cancelled");
downloads.remove(num-1);
break;
case 6:
System.out.println("Enter download number to get detailed info: ");

num = s.nextInt();
Download temp = downloads.get(num-1);

String name = temp.getFileName(new URL(temp.getUrl()));
String elapsedTime = temp.getElapsedTime();
```

```
float progress = temp.getProgress();
String remainingTime = temp.getRemainingTime();
float speed = temp.getSpeed();
long size = temp.getSize();
String status;
if(temp.getStatus()==1) status = "PAUSED";
else if(temp.getStatus()==0) status = "Downloading";
else if(temp.getStatus()==2) status = "Complete";
else status = "N/A";

System.out.println("----------------------------------------------------------------
------------------------------------");
System.out.println("File Name      : " +name);
System.out.println("File Size      : " +size +" bytes");
System.out.println("Status         : " +status);
System.out.println("Progress       : " +progress +" %");
System.out.println("Elapsed Time   : " +elapsedTime);
System.out.println("Remaining Time : " +remainingTime);
System.out.println("Speed          : " +speed +" KB/S");
System.out.println("----------------------------------------------------------------
-------------------------------------");
break;
case 7:
return;
default:
System.out.println("Please enter a valid option!");
}
}
}
}
```

5.Results & Interpretation

1. **Download a New File:**
    - Choose option 1, enter a valid download link (e.g., jpeg/png/zip/pdf), and watch your download start.
2. **Show Ongoing/Paused Downloads:**
    - Option 2 displays a list of ongoing and paused downloads.
3. **Pause a Download:**
    - Select option 3, enter the download number, and pause an ongoing download.
4. **Resume a Download:**
    - Choose option 4, specify the download number, and resume a paused download.
5. **Cancel a Download:**
    - Pick option 5, enter the download number, and cancel a download.
6. **Get Detailed Info of a Download:**
    - Select option 6, input the download number, and view detailed information about a download.
7. **Exit:**
    - Choose option 7 to exit the program.



kali@Maheshwar: /mnt/c/Use

```
┌──(kali㉿Maheshwar)-[/mnt/c/Users/hp/OneDrive/Desktop/IDM-JAVA]
└─$ java Main
               Welcome to IDM - Internet Download Manager
---------------------------------MENU:-----------------------------------
1 : Download a new file
2 : Show a list of currently ongoing/paused downloads
3 : Pause a download
4 : Resume a download
5 : Cancel a download
6 : Get detailed info of a download
7 : Exit
---------------------------------------------------------------------------
Enter your choice: 1
Enter valid download link [Ex:jpeg/png/zip/pdf] :http://212.183.159.230/1GB.zip
Connected to the server, Response: Partial Content
Download has started
Enter your choice: 2
1 : 1GB.zip Downloading
Enter your choice: 3
Enter the download number to pause: 1
Download 1GB.zip paused
Enter your choice: 4
Enter the download number to resume: 1
Download  for 1GB.zip resumed
Enter your choice: Connected to the server, Response: Partial Content
6
Enter download number to get detailed info:

---------------------------------------------------------
File Name               : 1GB.zip
File Size               : 1073741824 bytes
Status                  : Downloading
Progress                : 3.7655764 %
Elapsed Time    : 00:00:26
Remaining Time  : 00:08:31
Speed                   : 1971.3417 KB/S
---------------------------------------------------------
Enter your choice: 7
```

6. Conclusion

The development of a console-based Internet Download Manager (IDM) using Java presents a successful integration of core programming concepts to create a versatile and efficient tool for managing file downloads. This project has provided valuable insights into the implementation of networking protocols, multithreading, and user interface design within a console environment. Through the use of Java, we have demonstrated the capability to handle concurrent downloads, ensuring optimal utilization of available network bandwidth. The project's reliance on a console interface showcases the adaptability of Java to different application scenarios, offering a lightweight and user-friendly solution for download management. Despite the absence of a graphical user interface (GUI), the console-based IDM has proven to be a reliable and resource-efficient option for users who prioritize functionality over visual aesthetics. The project's successful implementation underscores the significance of careful design and systematic programming in creating a robust and user-friendly application. In the future, further enhancements could be explored, such as the incorporation of additional features, improved error handling, and compatibility with a broader range of protocols. Overall, this project serves as a testament to the capabilities of Java in creating powerful and functional software tools even within a console-based environment.

7.References

[1] Computer Networks, Andrew S. Tannenbaum,4th Edition, Pearson India.

[2] Java Network Programming by Harold, O'Reilly (Shroff Publishers).

[3] https://en.m.wikipedia.org/wiki/Internet_Download_Manager

[4]https://www.baeldung.com/java-download-file