ConsoleIDM.java

import java.io.\*;

import java.net.\*;

import java.nio.file.\*;

import java.util.Date;

public class ConsoleIDM {

public static volatile boolean paused = false;

public static void main(String args[]) throws Exception {

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("\* Welcome to Console IDM \*");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

// Menu-driven interface

while (true) {

System.out.println("\n1. Download File");

System.out.println("2. History");

System.out.println("3. Exit");

System.out.print("\nEnter your choice: ");

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

int choice = Integer.parseInt(reader.readLine());

switch (choice) {

case 2:

DBI.getHistory();

break;

case 1:

if (args.length == 0) {

System.out.println("Please provide the URLs as command-line arguments.");

} else {

for (String url : args) {

downloadFile(url);

}

}

break;

case 3:

System.out.println("Exiting...");

System.exit(0);

break;

default:

System.out.println("Invalid choice. Please enter a valid option.");

}

}

}

// Method to download a file from a given URL

private static void downloadFile(String urlToDownload) {

try {

URL u = new URL(urlToDownload);

URLConnection uc = u.openConnection();

// Display information about the file to be downloaded

System.out.println("\nDownloading file from:\n" + urlToDownload);

System.out.println("------------------------------------------------------");

System.out.println("Date: " + new Date(uc.getDate()));

System.out.println("Content-Type: " + uc.getContentType());

System.out.println("Expires: " + uc.getExpiration());

System.out.println("Last-Modified: " + new Date(uc.getLastModified()));

long len = uc.getContentLengthLong();

System.out.println("Content-Length: " + len);

// Check if the file size is greater than 0

if (len > 0) {

// Extract file name from URL

Path path = Paths.get(u.getPath());

String fileName = path.getFileName().toString();

int numThreads = 4; // Number of threads to use

long partSize = len / numThreads; // Calculate part size

// Create and start threads for downloading parts of the file

DownloadThread[] threads = new DownloadThread[numThreads];

for (int i = 0; i < numThreads; i++) {

long startByte = i \* partSize;

long endByte = (i == numThreads - 1) ? len : (i + 1) \* partSize - 1;

threads[i] = new DownloadThread(urlToDownload, fileName, (int) startByte, (int) endByte); // Casting long to int

threads[i].start();

}

// Monitor progress and allow pausing/resuming

long totalBytesDownloaded = 0;

long startTime = System.currentTimeMillis();

long lastDownloadedBytes = 0; // Keep track of last downloaded bytes for calculating speed when paused

while (totalBytesDownloaded < len) {

totalBytesDownloaded = 0;

for (int i = 0; i < numThreads; i++) {

totalBytesDownloaded += threads[i].getDownloadedBytes();

}

// Calculate download speed

long currentTime = System.currentTimeMillis();

long elapsedTime = currentTime - startTime;

double downloadSpeed = (totalBytesDownloaded - lastDownloadedBytes) / (double) elapsedTime \* 1000; // Bytes per second

downloadSpeed = downloadSpeed / 1024; // Kilobytes per second

lastDownloadedBytes = totalBytesDownloaded; // Update last downloaded bytes

printProgressBar(totalBytesDownloaded, len, downloadSpeed);

Thread.sleep(1000); // Update progress every second

// Check for user input to pause/resume download

if (System.in.available() > 0) {

char input = (char) System.in.read();

System.in.skip(System.in.available()); // Clear input buffer

if (input == 'p') {

paused = true;

System.out.println("\nDownload paused. Press 'r' to resume...");

continue; // Skip remaining logic and wait for next input

} else if (input == 'r') {

paused = false;

synchronized (ConsoleIDM.class) {

ConsoleIDM.class.notifyAll();

}

System.out.println("\nResuming download...");

startTime = System.currentTimeMillis(); // Reset start time to maintain accurate speed calculation

} else {

System.out.println("\nInvalid input. Press 'p' to pause or 'r' to resume.");

}

}

}

// Log downloaded file

logDownloadedFile(fileName, len, urlToDownload);

System.out.println("\nDownload completed. Saved as: " + fileName);

System.out.println("------------------------------------------------------");

} else {

System.out.println("\nNo content available for download.");

}

} catch (Exception e) {

e.printStackTrace();

}

}

// Method to print progress bar and download speed

private static void printProgressBar(long current, long total, double downloadSpeed) {

int progress = (int) (100.0 \* current / total);

System.out.printf("\r[%d%%] [%d/%d] Download Speed: %.2f KB/s", progress, current, total, downloadSpeed);

}

// Method to log downloaded file

public static void logDownloadedFile(String fileName, long size, String urlToDownload) {

try {

FileWriter writer = new FileWriter("downloaded\_files.txt", true);

BufferedWriter bufferWriter = new BufferedWriter(writer);

bufferWriter.write(fileName + " - " + size + " bytes - " + new Date() + "\n");

bufferWriter.close();

// Save download history to database

DBI.saveDownloadHistory(urlToDownload, fileName, size, new Date(), new Date(), true);

} catch (IOException e) {

e.printStackTrace();

}

}

}

DBI.java

import java.io.\*;

import java.util.\*;

import java.sql.\*;

class DBI {

// Method to save download history to database

public static void saveDownloadHistory(String url, String fileName, long size, java.util.Date startTime, java.util.Date endTime, boolean success) {

Connection con = null;

Statement st = null;

try {

Class.forName("com.mysql.jdbc.Driver");

con = DriverManager.getConnection("jdbc:mysql://localhost:3306/idm?characterEncoding=utf8", "root", "");

st = con.createStatement();

// Create the SQL query with proper formatting of values

String sql = "INSERT INTO download (url, file\_name, size, start\_time, end\_time, success) VALUES (" +

"'" + url + "', '" + fileName + "', " + size + ", '" +

new java.sql.Timestamp(startTime.getTime()) + "', '" +

new java.sql.Timestamp(endTime.getTime()) + "', " + (success ? 1 : 0) + ")";

// Execute the SQL query

int rowsAffected = st.executeUpdate(sql);

// Check if the insertion was successful

if (rowsAffected > 0) {

System.out.println("\nDownload history stored successfully.");

} else {

System.out.println("Failed to store download history. No rows affected.");

}

} catch (ClassNotFoundException | SQLException e) {

e.printStackTrace();

System.out.println("Failed to store download history due to exception: " + e.getMessage());

} finally {

// Close the connection and statement

try {

if (st != null) {

st.close();

}

if (con != null) {

con.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

public static void getHistory() {

Connection con = null;

Statement st = null;

ResultSet rs = null;

try {

Class.forName("com.mysql.jdbc.Driver");

con = DriverManager.getConnection("jdbc:mysql://localhost:3306/idm?characterEncoding=utf8", "root", "");

st = con.createStatement();

String str = "select \* from download";

rs = st.executeQuery(str);

System.out.println("URL | File Name | Size | Start Time | End Time | Success");

System.out.println("------------------------------+--------------------+-----------+-------------------------+-------------------------+--------");

while (rs.next()) {

System.out.println(rs.getString("url") + " | " + rs.getString("file\_name") + " | " + rs.getLong("size") +

" " + rs.getTimestamp("start\_time") + " | " + rs.getTimestamp("end\_time") + " | " + rs.getBoolean("success"));

System.out.println("------------------------------+--------------------+-----------+-------------------------+-------------------------+--------");

}

} catch (Exception e) {

System.out.println("Error " + e);

} finally {

// Close the connection, statement, and result set

try {

if (rs != null) {

rs.close();

}

if (st != null) {

st.close();

}

if (con != null) {

con.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

DownloadThread.java

import java.io.\*;

import java.net.\*;

public class DownloadThread extends Thread {

private String urlToDownload;

private String fileName;

private long startByte;

private long endByte;

private long downloadedBytes;

public DownloadThread(String urlToDownload, String fileName, long startByte, long endByte) {

this.urlToDownload = urlToDownload;

this.fileName = fileName;

this.startByte = startByte;

this.endByte = endByte;

}

public void run() {

try {

URL u = new URL(urlToDownload);

HttpURLConnection uc = (HttpURLConnection) u.openConnection();

uc.setRequestProperty("Range", "bytes=" + startByte + "-" + endByte);

InputStream in = uc.getInputStream();

RandomAccessFile raf = new RandomAccessFile(fileName, "rw");

raf.seek(startByte);

byte[] buffer = new byte[1024];

int bytesRead;

while ((bytesRead = in.read(buffer)) != -1) {

synchronized (ConsoleIDM.class) {

while (ConsoleIDM.paused) {

try {

ConsoleIDM.class.wait();

} catch (InterruptedException e) {

e.printStackTrace(); // Handle interruption as needed

}

}

}

raf.write(buffer, 0, bytesRead);

downloadedBytes += bytesRead;

}

raf.close();

in.close();

} catch (IOException e) {

e.printStackTrace();

}

}

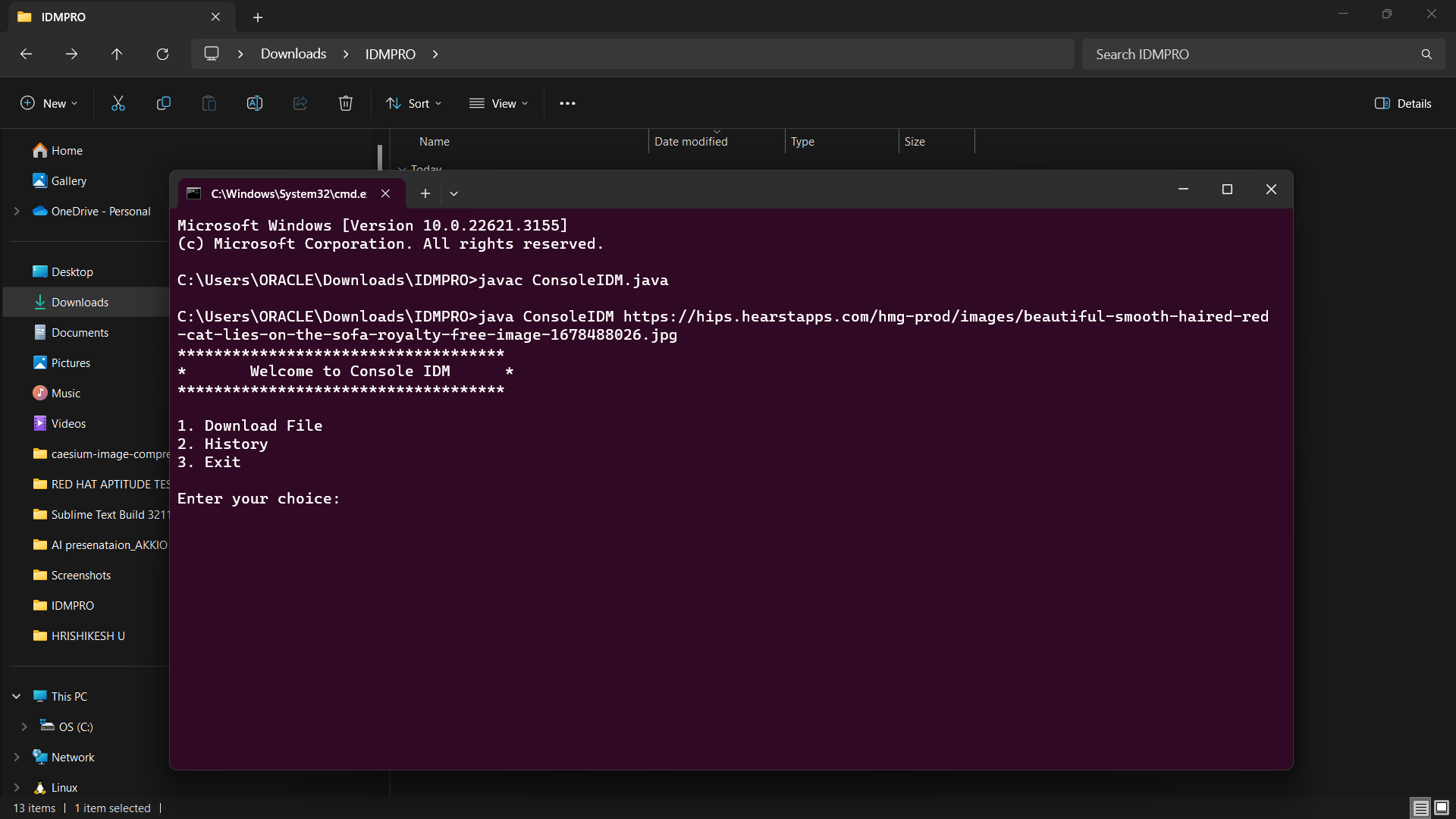
public long getDownloadedBytes() {

return downloadedBytes;

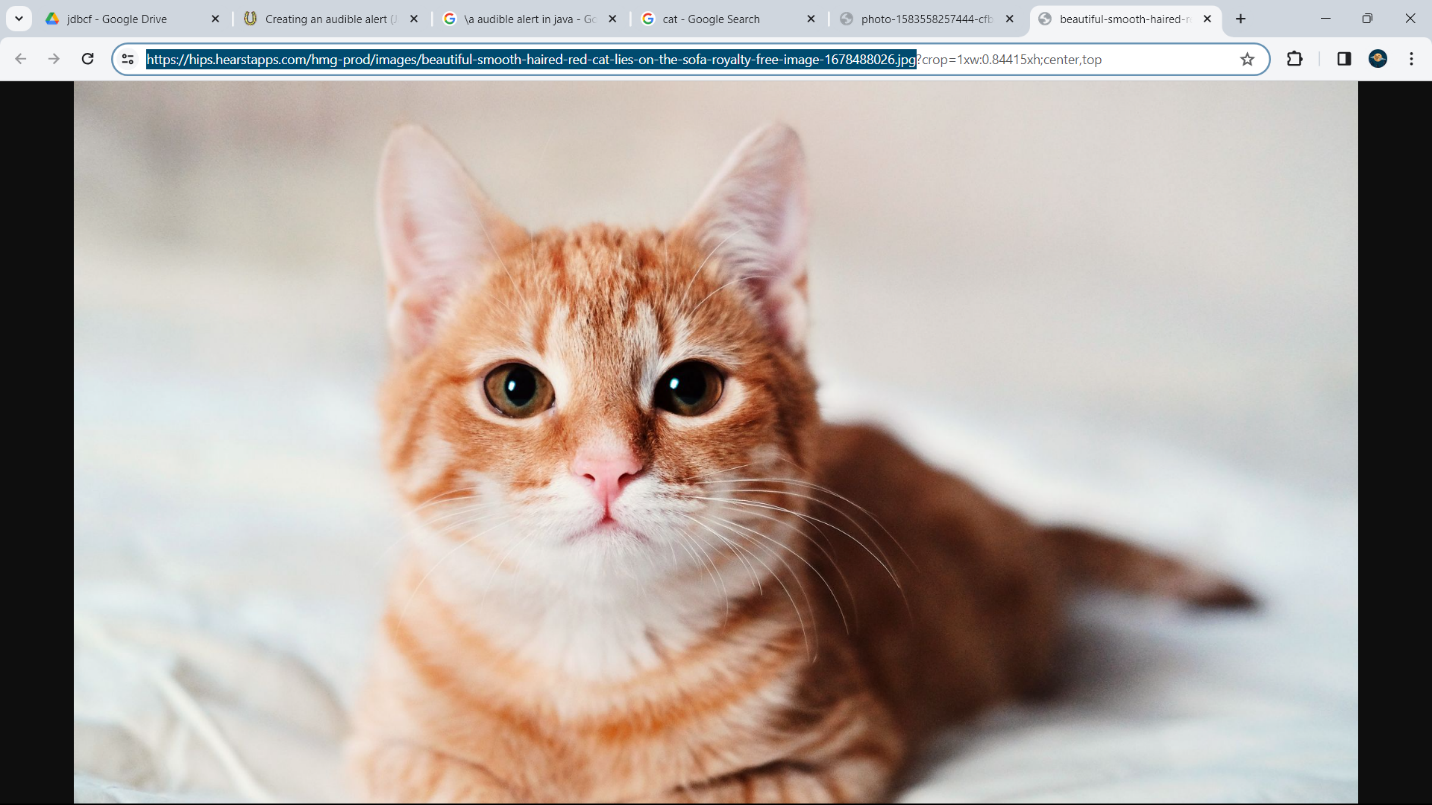
}

}

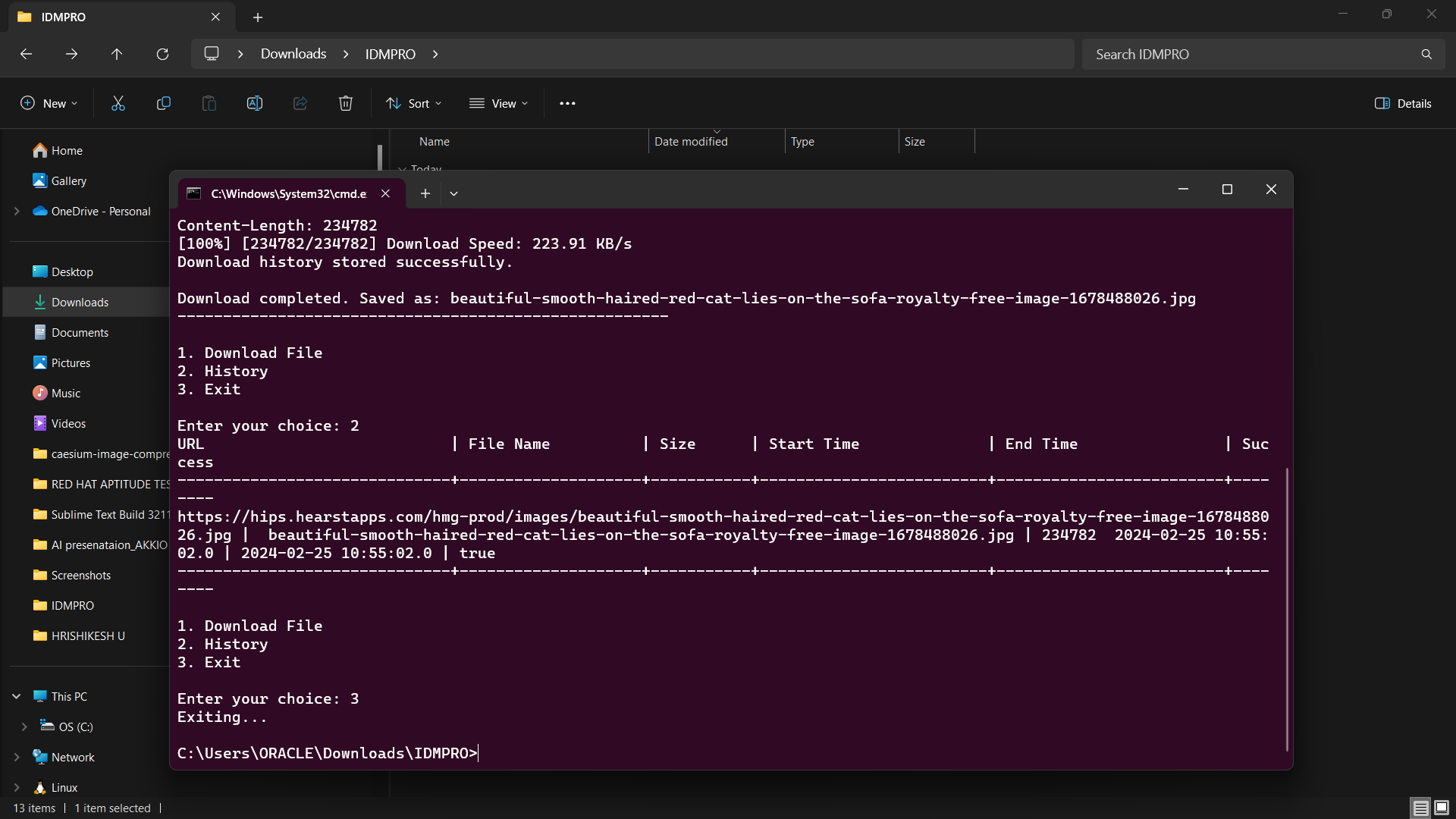
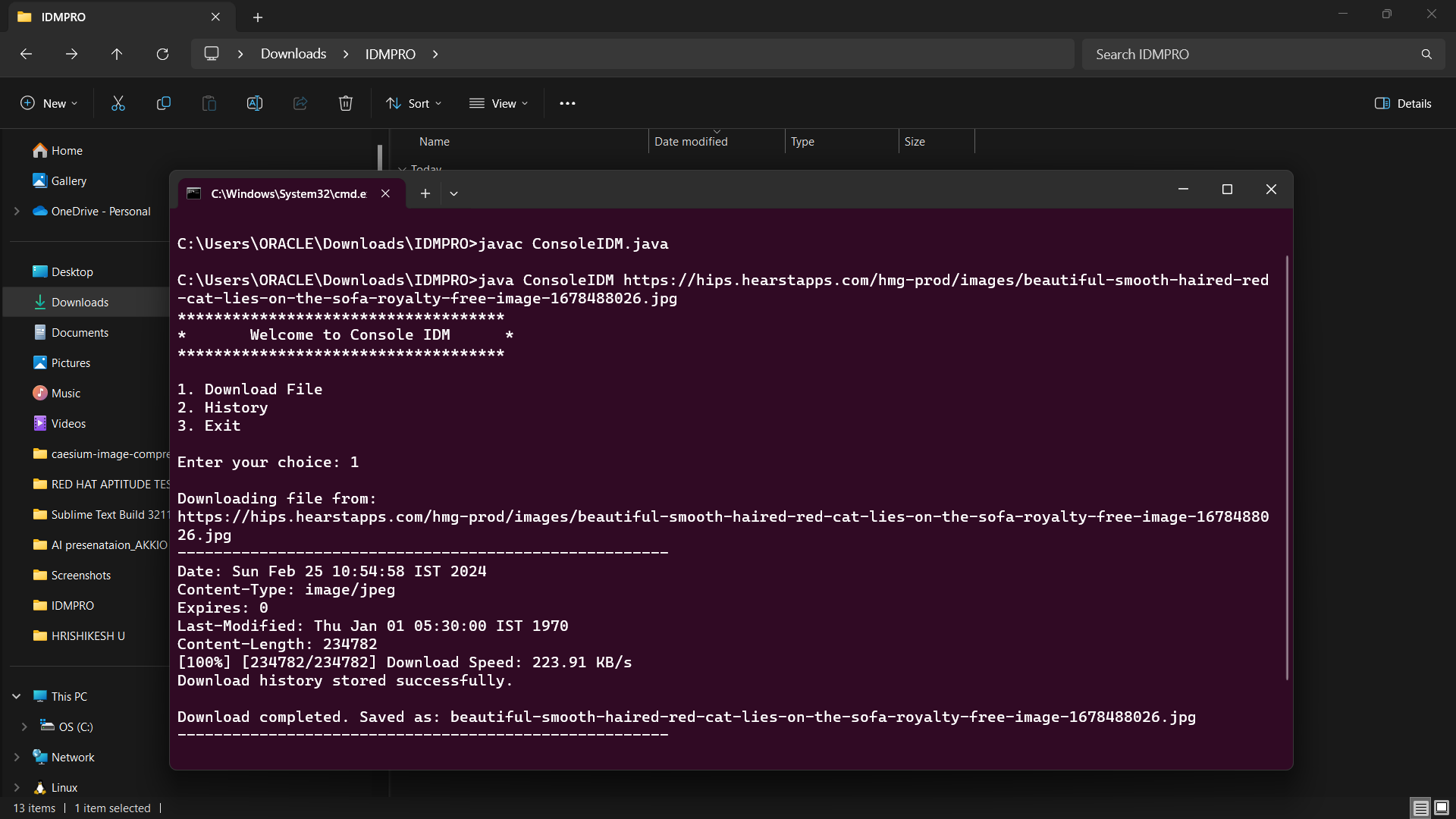
STEP 1:



STEP 2:



STEP 3:



STEP 4:

