

```
import java.io.*;
import java.util.zip.*;

/**
 * Command line program to copy a file to another directory.
 * @author Marco Schmidt
 */
public class CopyFile {
    // constant values for the override option
    public static final int OVERWRITE_ALWAYS = 1;
    public static final int OVERWRITE_NEVER = 2;
    public static final int OVERWRITE_ASK = 3;

    // program options initialized to default values
    private static int bufferSize = 4 * 1024;
    private static boolean clock = true;
    private static boolean copyOriginalTimestamp = true;
    private static boolean verify = true;
    private static int override = OVERWRITE_ALWAYS;

    public static Long copyFile(File srcFile, File destFile)
        throws IOException {
        InputStream in = new FileInputStream(srcFile);
        OutputStream out = new FileOutputStream(destFile);
        long millis = System.currentTimeMillis();
        CRC32 checksum = null;
        if (verify) {
            checksum = new CRC32();
            checksum.reset();
        }
        byte[] buffer = new byte[bufferSize];
        int bytesRead;
        while ((bytesRead = in.read(buffer)) >= 0) {
            if (verify) {
                checksum.update(buffer, 0, bytesRead);
            }
            out.write(buffer, 0, bytesRead);
        }
        out.close();
        in.close();
        if (clock) {
            millis = System.currentTimeMillis() - millis;
            System.out.println("Second(s): " + (millis/1000L));
        }
        if (verify) {
            return new Long(checksum.getValue());
        } else {
            return null;
        }
    }

    public static Long createChecksum(File file) throws IOException {
        long millis = System.currentTimeMillis();
    }
}
```

```

InputStream in = new FileInputStream(file);
CRC32 checksum = new CRC32();
checksum.reset();
byte[] buffer = new byte[bufferSize];
int bytesRead;
while ((bytesRead = in.read(buffer)) >= 0) {
    checksum.update(buffer, 0, bytesRead);
}
in.close();
if (clock) {
    millis = System.currentTimeMillis() - millis;
    System.out.println("Second(s): " + (millis/1000L));
}
return new Long(checksum.getValue());
}

/**
 * Determine if data is to be copied to given file.
 * Take into consideration override option and
 * ask user in case file exists and override option is ask.
 * @param file File object for potential destination file
 * @return true if data is to be copied to file, false if not
 */
public static boolean doCopy(File file) {
    boolean exists = file.exists();
    if (override == OVERWRITE_ALWAYS || !exists) {
        return true;
    } else
    if (override == OVERWRITE_NEVER) {
        return false;
    } else
    if (override == OVERWRITE_ASK) {
        return readYesNoFromStandardInput("File exists. " +
            "Overwrite (y/n)?");
    } else {
        throw new InternalError("Program error. Invalid " +
            "value for override: " + override);
    }
}

public static void main(String[] args) throws IOException {
    // make sure there are exactly two arguments
    if (args.length != 2) {
        System.err.println("Usage: CopyFile SRC-FILE-NAME DEST-DIR-NAME");
        System.exit(1);
    }
    // make sure the source file is indeed a readable file
    File srcFile = new File(args[0]);
    if (!srcFile.isFile() || !srcFile.canRead()) {
        System.err.println("Not a readable file: " + srcFile.getName());
        System.exit(1);
    }
    // make sure the second argument is a directory

```

```

File destDir = new File(args[1]);
if (!destDir.isDirectory()) {
    System.err.println("Not a directory: " + destDir.getName());
    System.exit(1);
}
// create File object for destination file
File destFile = new File(destDir, srcFile.getName());

// check if copying is desired given overwrite option
if (!doCopy(destFile)) {
    return;
}

// copy file, optionally creating a checksum
Long checksumSrc = copyFile(srcFile, destFile);

// copy timestamp of last modification
if (copyOriginalTimestamp) {
    if (!destFile.setLastModified(srcFile.lastModified())) {
        System.err.println("Error: Could not set " +
            "timestamp of copied file.");
    }
}

// optionally verify file
if (verify) {
    System.out.print("Verifying destination file...");
    Long checksumDest = createChecksum(destFile);
    if (checksumSrc.equals(checksumDest)) {
        System.out.println(" OK, files are equal.");
    } else {
        System.out.println(" Error: Checksums differ.");
    }
}
}

/**
 * Print a message to standard output and read lines from
 * standard input until yes or no (y or n) is entered.
 * @param message informative text to be answered by user
 * @return user answer, true for yes, false for no.
 */
public static boolean readYesNoFromStandardInput(String message) {
    System.out.println(message);
    String line;
    BufferedReader in = new BufferedReader(new InputStreamReader(
        System.in));
    Boolean answer = null;
    try
    {
        while ((line = in.readLine()) != null) {
            line = line.toLowerCase();
            if ("y".equals(line) || "yes".equals(line)) {

```

```
        answer = Boolean.TRUE;
        break;
    }
    else
    if ("n".equals(line) || "no".equals(line)) {
        answer = Boolean.FALSE;
        break;
    }
    else
    {
        System.out.println("Could not understand answer (\\" +
            line + "\\"). Please use y for yes or n for no.");
    }
}
if (answer == null) {
    throw new IOException("Unexpected end of input from stdin.");
}
in.close();
return answer.booleanValue();
}
catch (IOException ioe)
{
    throw new InternalError(
        "Cannot read from stdin or write to stdout.");
}
}
}
```