Java I/O, Files, Streams and Object Serialization

1. Write a Java program that creates a text file named "mydata.txt" and writes the string "Hello, Java Streams!" into it using a FileOutputStream or FileWriter.
2. Create a Java program that reads the content of "mydata.txt" (created in the previous exercise) and prints it to the console using a FileInputStream or FileReader
3. Modify the file writing program to append new text (e.g., "This is appended text.") to "mydata.txt" without overwriting the existing content.
4. Write a Java program that copies the content of one text file (e.g., "source.txt") to another text file (e.g., "destination.txt") using BufferedInputStream and BufferedOutputStream for efficient copying.
5. Write a Java program to copy the contents of one binary file (e.g., an image or a PDF) to another using FileInputStream and FileOutputStream.
6. Write a program that reads a sequence of bytes from a file and writes them to another file, converting them to uppercase during the process.
7. Write a program that takes a text file as input and counts the number of characters, words, and lines in it.
8. Define a simple Person class with attributes like name and age. Make the Person class Serializable. Write a program to create a Person object and save it to a file using ObjectOutputStream. Write another program to read the Person object back from the file using ObjectInputStream and print its details.
9. Create a program that writes text to a file using a specific character encoding (e.g., UTF-8, ISO-8859-1) and then reads it back, ensuring correct character display.
10. Write a Java program that creates a new text file named "data.txt" in a specified directory. Then, write another program to delete this file.
11. Write a program that takes a file path as input and checks if it exists, if it's a file or a directory, and its read/write permissions
12. Write a program that takes a directory path as input and lists all files and subdirectories within it.