1. **Basic File Operations:**
    - Write a Java program to create a new file named `testfile.txt`. Check if the file was created successfully.
    - Use the `File` class to retrieve and print the absolute path of `testfile.txt`.
2. **Checking File Properties:**
    - Create a program to check whether `testfile.txt` is a file or a directory.
    - Verify if `testfile.txt` exists and output an appropriate message.
3. **Deleting Files:**
    - Write a program that deletes `testfile.txt` and prints whether the operation was successful.
4. **Buffered I/O:**

Implement a program using `BufferedWriter` to write the following lines to a file named `bufferedoutput.txt`:
mathematica

```
Line 1: Java file handling is efficient.
Line 2: BufferedWriter improves performance.
```

    - Read back the content of `bufferedoutput.txt` using `BufferedReader` and print it line by line.
5. **Serialization and Deserialization:**
    - Create a `Person` class that implements `Serializable`. The class should have `name` (String) and `age` (int) attributes.
    - Write a program to:
        - Serialize an object of `Person` (e.g., `new Person("Alice", 25)`) to a file named `person.ser`.
        - Deserialize the object from `person.ser` and print its attributes.
6. **Exception Handling:**

    - Modify one of the file operations to handle potential exceptions gracefully, such as `IOException` or `FileNotFoundException`.
7. **Exploring File Class Methods:**
    - Write a program to create a directory named `exampleDir`.
    - Within `exampleDir`, create a new file named `exampleFile.txt`.
    - Check whether `exampleDir` is a directory and `exampleFile.txt` is a file.
8. **Stream vs. File Class:**
    - Discuss the advantages of using streams over the `File` class for handling large files. Include an example where `BufferedReader` is more efficient than `FileReader`.
9. **Large File Handling:**
    - Simulate processing a large file using `BufferedReader`. Create a program that reads a file line by line and counts the total number of lines.
10. **Exploring Object Streams:**
    - Extend the serialization task by adding another attribute to the `Person` class (e.g., `email`).
    - Serialize and deserialize an updated `Person` object with the new attribute. Verify that all attributes are restored correctly.