Experiment - 5

Student Name: Keshav Chandra Kumar **UID:** 22BCS16461
Branch: CSE **Section/Group:** 642/A
Semester: 6th **Date of Performance:** 21.02.25
Subject Name: JAVA **Subject Code:** 22CSH-359

1. Aim: To develop Java programs that demonstrate the use of autoboxing, serialization, file handling, and efficient data processing and management, ensuring optimized performance and structured data handling.

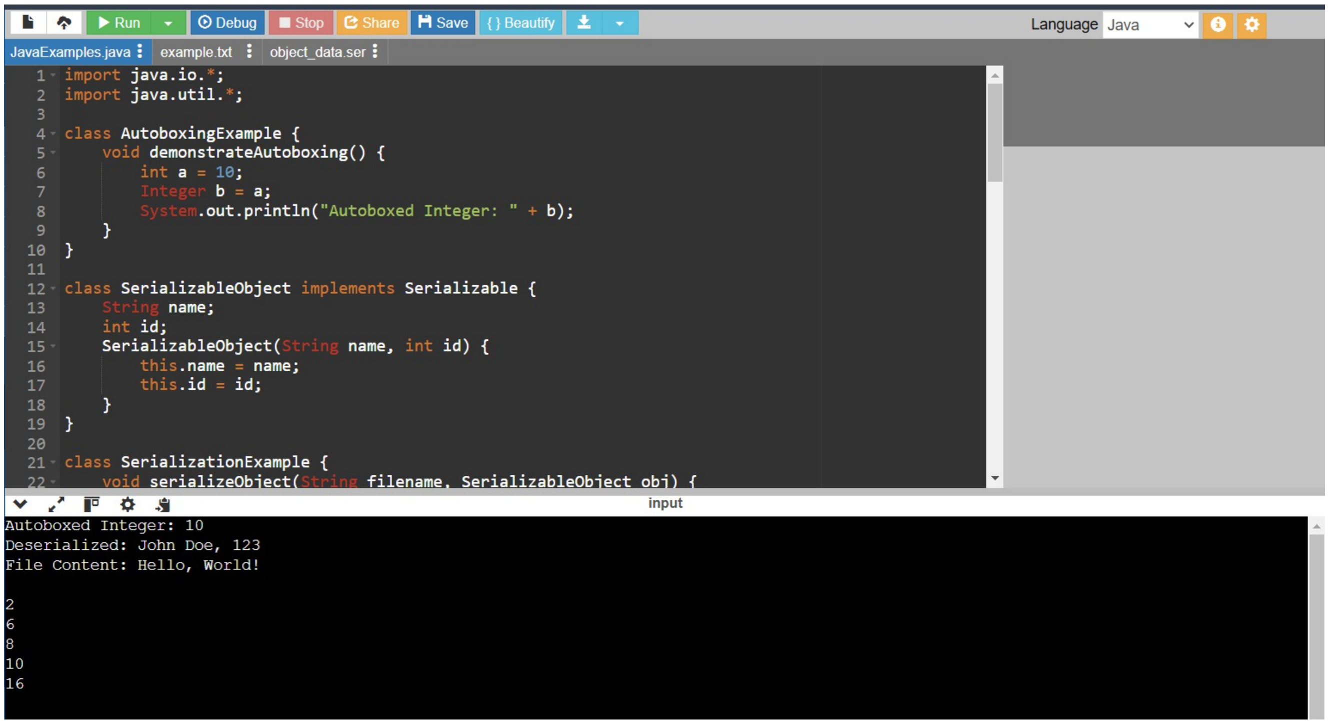
2. Objective:

- **Utilize Autoboxing and Unboxing** – Implement Java programs that efficiently convert between primitive types and their corresponding wrapper classes.
- **Implement Serialization** – Develop Java applications that serialize and deserialize objects to facilitate data storage and transmission.
- **Handle Files Effectively** – Read, write, and manipulate files using Java's File I/O APIs for persistent data management.
- **Optimize Data Processing** – Implement efficient algorithms and data structures for handling large datasets with minimal performance overhead.

3. Implementation/Code:

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

class AutoboxingExample {
    void demonstrateAutoboxing()
        { int a = 10;
        Integer b = a;
        System.out.println("Autoboxed Integer: " + b);
    }
}

class SerializableObject implements Serializable
    { String name;
    int id;
    SerializableObject(String name, int id)
        { this.name = name;
        this.id = id;
    }
}

class SerializationExample {
    void serializeObject(String filename, SerializableObject obj) {
        try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(filename)))
{
            out.writeObject(obj);
         } catch (IOException e)
          { e.printStackTrace();
        }
    }

    SerializableObject deserializeObject(String filename) {
        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(filename)))
            { return (SerializableObject) in.readObject();
        } catch (IOException | ClassNotFoundException e)
            { e.printStackTrace();
        }
        return null;
    }
}

class FileHandlingExample {
    void writeFile(String filename, String content) {
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(filename)))
            { writer.write(content);
        } catch (IOException e)
            { e.printStackTrace();
        }
    }

    String readFile(String filename)
        { StringBuilder content = new
        StringBuilder();
        try (BufferedReader reader = new BufferedReader(new FileReader(filename))) {
            String line;
            while ((line = reader.readLine()) != null)
                { content.append(line).append("\n");
            }
        } catch (IOException e)
            { e.printStackTrace();
        }
        return content.toString();
    }
}

class DataProcessingExample {
    void processData(List<Integer> numbers) {
        numbers.stream().map(n -> n * 2).sorted().forEach(System.out::println);
    }
}

public class JavaExamples {
    public static void main(String[] args) {
        new AutoboxingExample().demonstrateAutoboxing();
        SerializationExample serializationExample = new SerializationExample();
        SerializableObject obj = new SerializableObject("John Doe", 123);
        String filename = "object_data.ser";
        serializationExample.serializeObject(filename, obj);
        SerializableObject deserializedObj = serializationExample.deserializeObject(filename);
        System.out.println("Deserialized: " + deserializedObj.name + ", " + deserializedObj.id);
        FileHandlingExample fileExample = new FileHandlingExample();
        String file = "example.txt";
        fileExample.writeFile(file, "Hello, World!");
        System.out.println("File Content: " + fileExample.readFile(file));
        List<Integer> numbers = Arrays.asList(5, 3, 8, 1, 4);
        new DataProcessingExample().processData(numbers);
    }
}
```

4. Output:



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

class AutoboxingExample {
    void demonstrateAutoboxing() {
        int a = 10;
        Integer b = a;
        System.out.println("Autoboxed Integer: " + b);
    }
}

class SerializableObject implements Serializable {
    String name;
    int id;
    SerializableObject(String name, int id) {
        this.name = name;
        this.id = id;
    }
}

class SerializationExample {
    void serializeObject(String filename, SerializableObject obj) {
```

```
Autoboxed Integer: 10
Deserialized: John Doe, 123
File Content: Hello, World!

2
6
8
10
16
```

5. Learning Outcome:

After implementing these programs, you will:

1. Understand autoboxing, which simplifies working with primitive types and wrapper classes.
2. Gain knowledge of serialization, allowing objects to be stored and retrieved efficiently.
3. Learn file handling, enabling interaction with external files for persistent data storage.
4. Develop efficient data processing skills using Java Streams, making data operations faster and more readable.