## 01. Introduction of the scope

Serialization and deserialization are one of the two basic fundamental processes of computer science. Serialization is the process of complicated data structures or objects converted into formats that are easily storable or easily transmittable. Deserialization is the process of reversing this transmitted or stored data into its original form and complexity.

## 02. Explain the benefits that can be derived from the scope.

**Persistence:** Serialized objects can be saved to files or databases, enabling long-term storage and retrieval.

**Cloning:** Serialization can be used to create deep copies of objects, ensuring that the entire object graph is duplicated.

**Caching:** Serialized objects can be stored in caches, this improves the performance of the application since it reduces the need to create new objects.

**Data Transfer:** It enables sending complex data structures across a network, which is very useful for client-server applications, web services, and distributed systems.

**Language Independence:** Some serialization formats (JSON, XML) can be read by multiple programming languages, this provides the interoperability between different systems.

## 03. Explain the mechanism of the scope.

### Process of Serialization:

Conversion to Byte Stream: Serialization is the process of converting the state of an object into a sequence of bytes. This stream can be stored in a file, database, or transmitted across a network.

**Serializable Interface:** In many programming languages (like Java), classes need to implement a specific interface (e.g., Serializable in Java) to be eligible for serialization.

**ObjectOutputStream:** The serialization process typically involves an ObjectOutputStream class (or similar functionality). This class provides methods to write the object's data fields, one by one, to the byte stream.

**Focus on Data:** Serialization primarily focuses on the object's data members and their values, essentially creating a snapshot of the object's state at that point in time.

### Process of Deserialization:

**Persistence:** Serialization allows objects to be stored and retrieved later, making data persistent.

**Data Transfer:** Serialized data can be easily transferred between applications or systems, promoting data exchange.

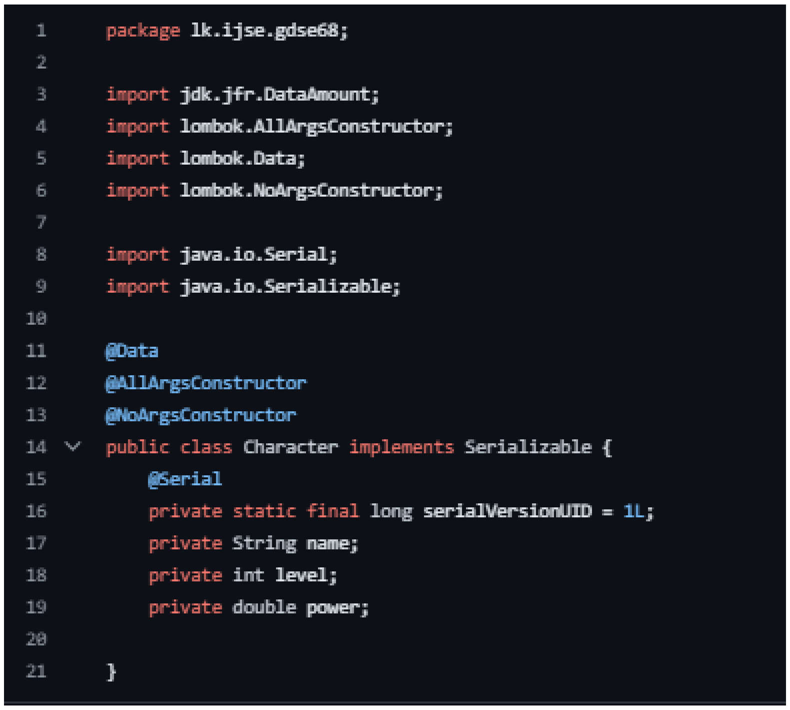
Platform Independence: In some cases, serialized data can be platform-independent, meaning it can be deserialized on different systems.

A new object is created using the extracted class information.The object's state is reconstructed using the extracted data.

Example Code Snippet-------

https://github.com/ashanravindu2/Serialization_and_Deserialization

Example



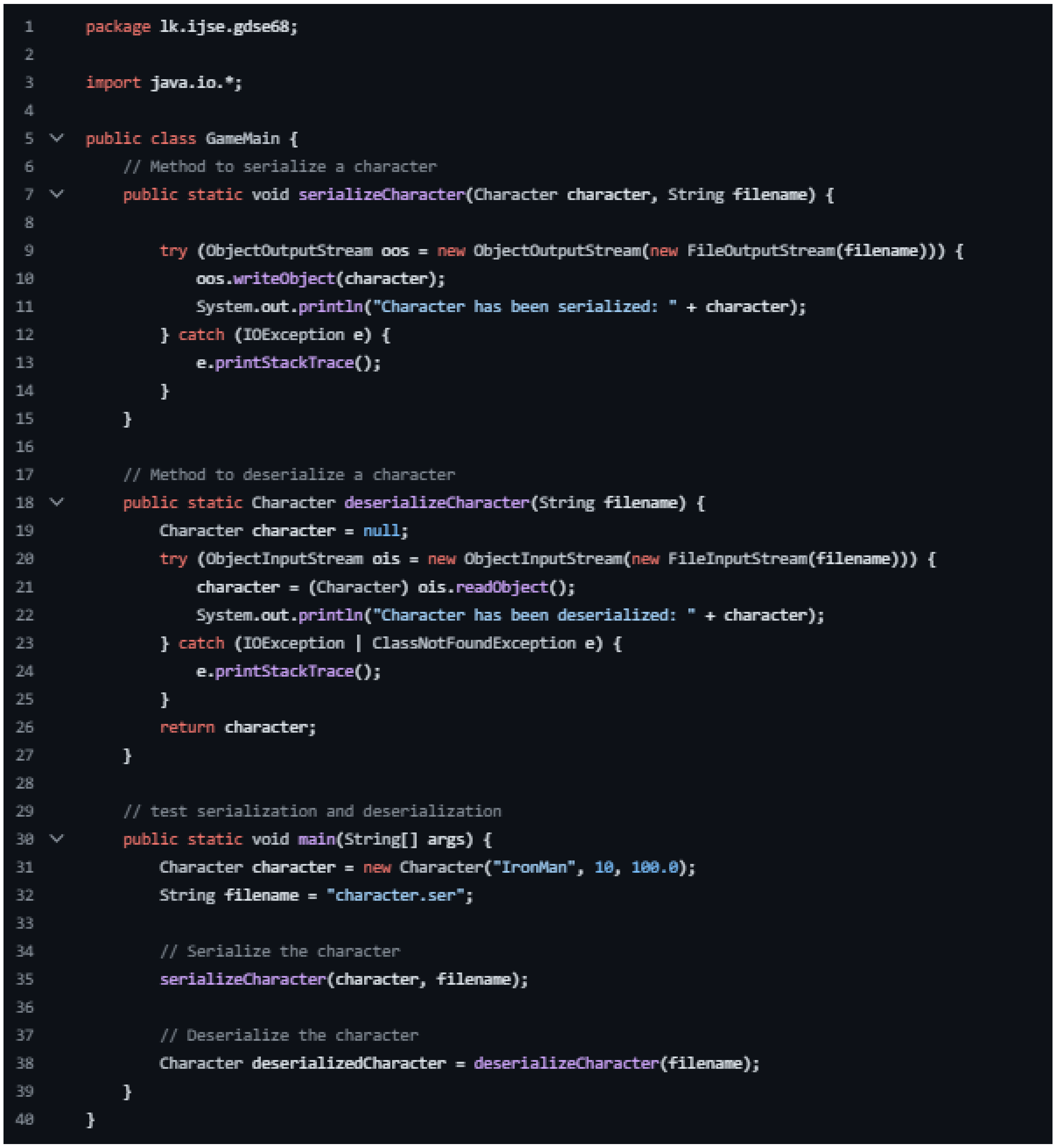
```java
package lk.ijse.gdse68;

import jdk.jfr.DataAmount;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.io.Serial;
import java.io.Serializable;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class Character implements Serializable {
    @Serial
    private static final long serialVersionUID = 1L;
    private String name;
    private int level;
    private double power;

}
```



```java
package lk.ijse.gdse68;

import java.io.*;

public class GameMain {
    // Method to serialize a character
    public static void serializeCharacter(Character character, String filename) {

        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(filename))) {
            oos.writeObject(character);
            System.out.println("Character has been serialized: " + character);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    // Method to deserialize a character
    public static Character deserializeCharacter(String filename) {
        Character character = null;
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(filename))) {
            character = (Character) ois.readObject();
            System.out.println("Character has been deserialized: " + character);
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
        return character;
    }

    // test serialization and deserialization
    public static void main(String[] args) {
        Character character = new Character("IronMan", 10, 100.0);
        String filename = "character.ser";

        // Serialize the character
        serializeCharacter(character, filename);

        // Deserialize the character
        Character deserializedCharacter = deserializeCharacter(filename);
    }
}
```

## Conclusion

With a little information about what Serialization and Deserialization are or why they matter, you may not realize how important they actually are within our daily lives. Ah, but this could not be further from reality (although some lucky people may never experience any of it). When you think about computers then think about their large chunky hardware; many people would not immediately associate power too. To most individuals, these machines which seem to perform magic remain a mystery-even today there isn't anyone who fully comprehends them! And if all that is still not enough for you, bear in mind that without such technical know-how many systems cannot function well.