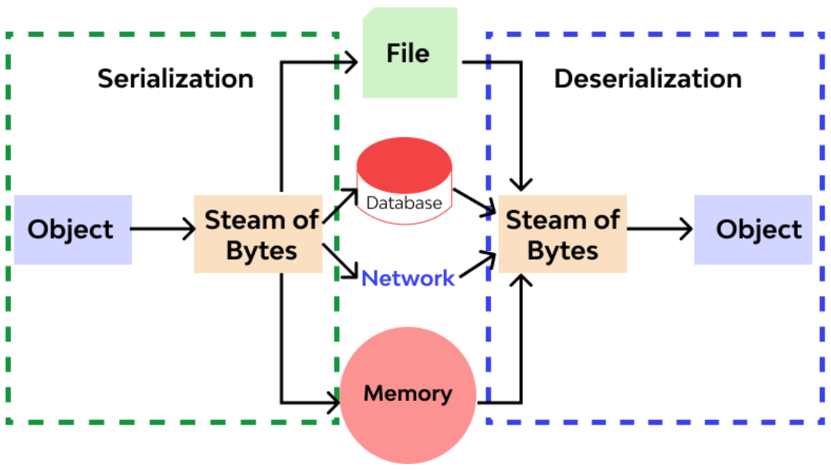
Serialization and deserialization in java

Serialization is a mechanism of converting the state of an object into a byte stream. Deserialization is the reverse process where the byte stream is used to recreate the actual Java object in memory.

Note: The byte stream created is platform independent. So, the object serialized on one platform can be deserialized on a different platform.



Why we need Serialization in java?

- To transfer objects through a network.
- To keep Java objects in memory.
- To save Java objects in files.

How to implement serialization in java?

1.For serializing the object, we call the writeObject() method of ObjectOutputStream, and for deserialization we call the readObject() method of ObjectInputStream class.

2.We must implement the Serializable interface for serializing the object.

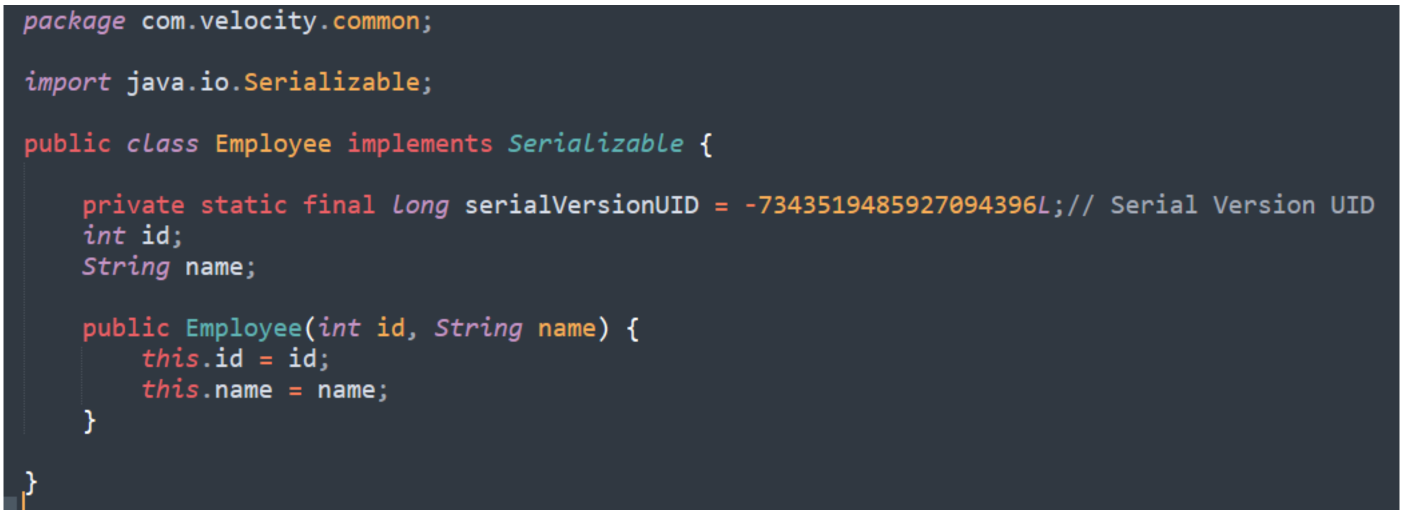
What is Serial Version UID?

Before the process of serialization begins, every serializable class/object gets associated with a unique identification number provided by the JVM of the host machine. This Unique ID is called Serial Version UID. This UID is used as an identification by the JVM of the receiving end to confirm that the same object is being DeSerialized at the receiving end.

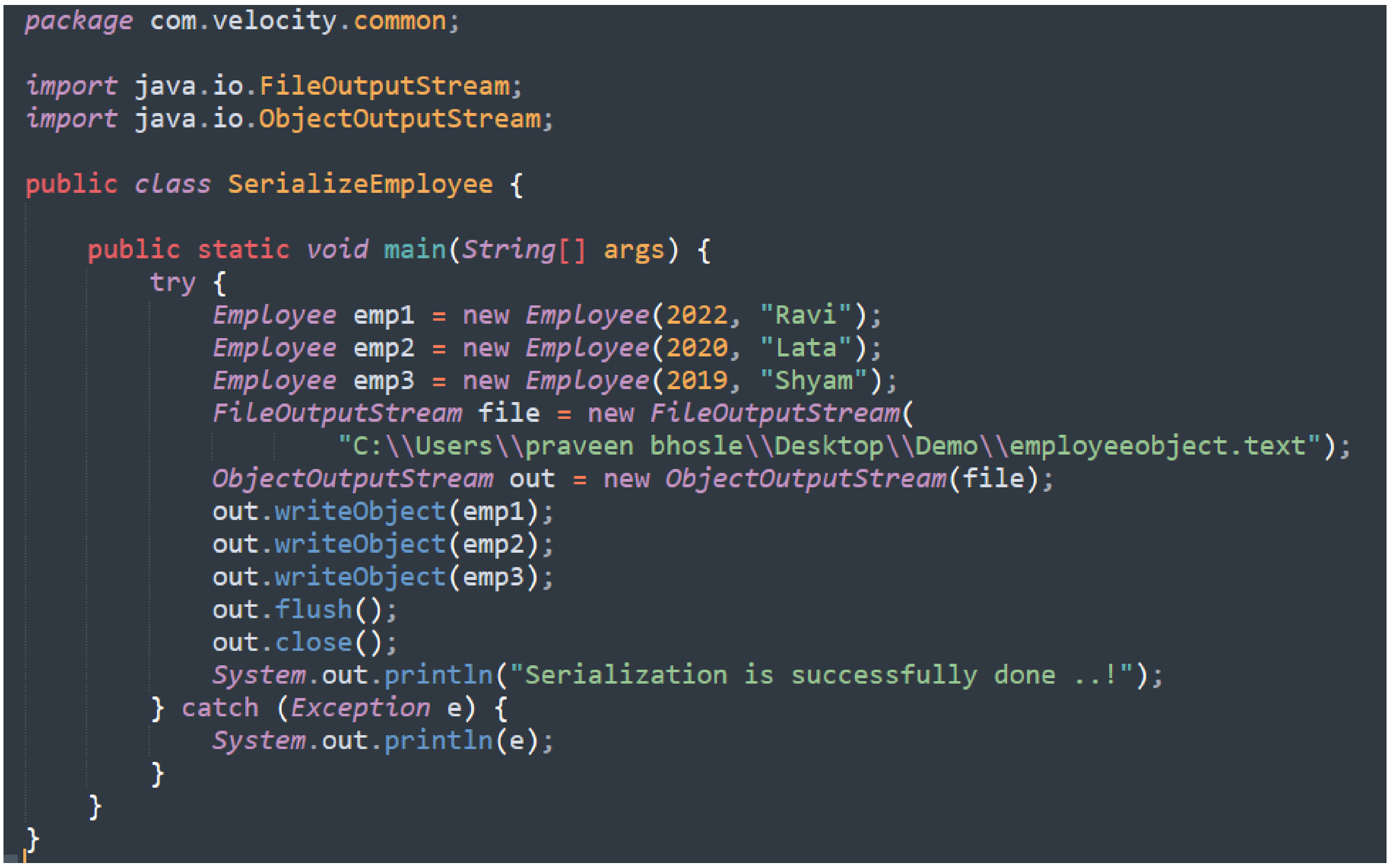
What is Transient Keyword?

Transient Keyword is a reserved keyword in Java. It is used as a variable modifier at the time of the Serialization process. Declaring a variable with Transient keyword avoids the variable from being Serialized.

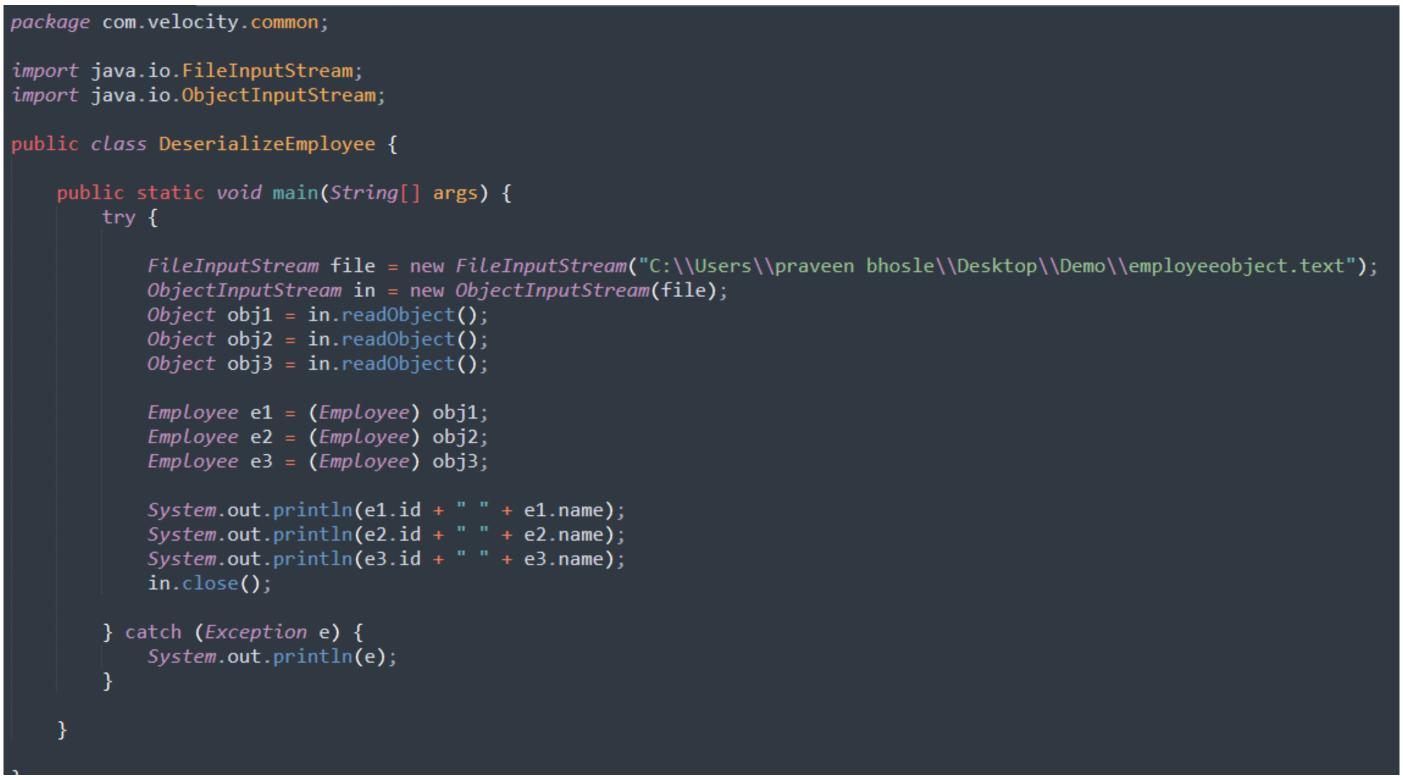
Example:1



```java
package com.velocity.common;

import java.io.Serializable;

public class Employee implements Serializable {

    private static final long serialVersionUID = -7343519485927094396L;// Serial Version UID
    int id;
    String name;

    public Employee(int id, String name) {
        this.id = id;
        this.name = name;
    }

}
```



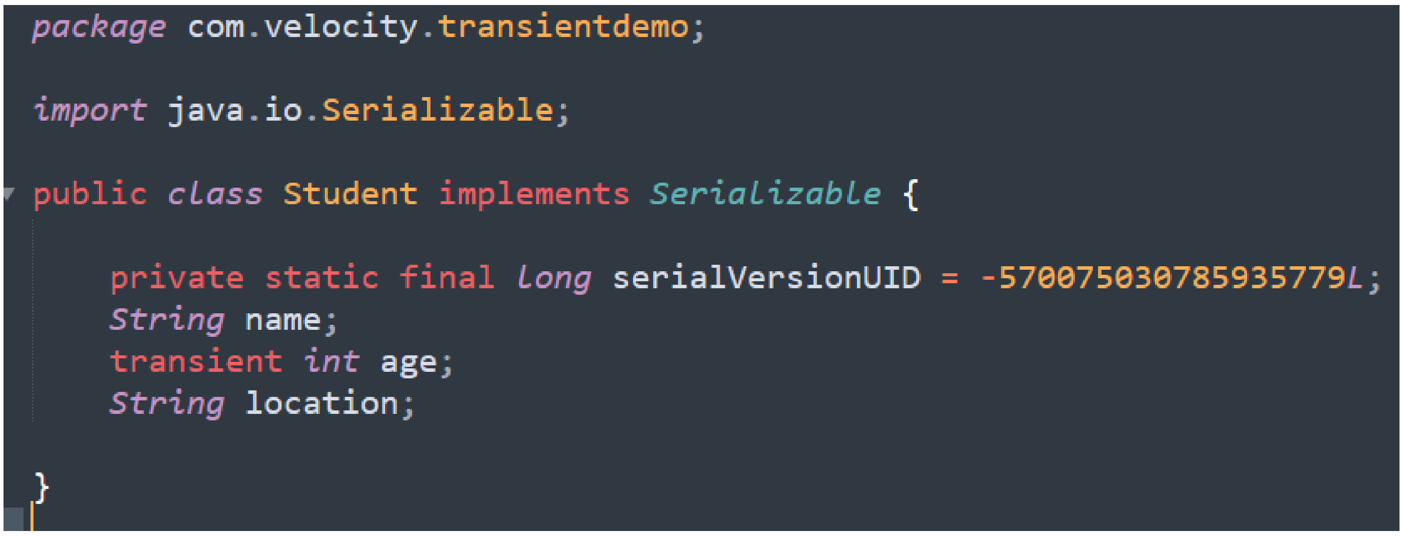
```java
package com.velocity.common;

import java.io.FileOutputStream;
import java.io.ObjectOutputStream;

public class SerializeEmployee {

    public static void main(String[] args) {
        try {
            Employee emp1 = new Employee(2022, "Ravi");
            Employee emp2 = new Employee(2020, "Lata");
            Employee emp3 = new Employee(2019, "Shyam");
            FileOutputStream file = new FileOutputStream(
                    "C:\\Users\\praveen bhosle\\Desktop\\Demo\\employeeobject.text");
            ObjectOutputStream out = new ObjectOutputStream(file);
            out.writeObject(emp1);
            out.writeObject(emp2);
            out.writeObject(emp3);
            out.flush();
            out.close();
            System.out.println("Serialization is successfully done ..!");
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```



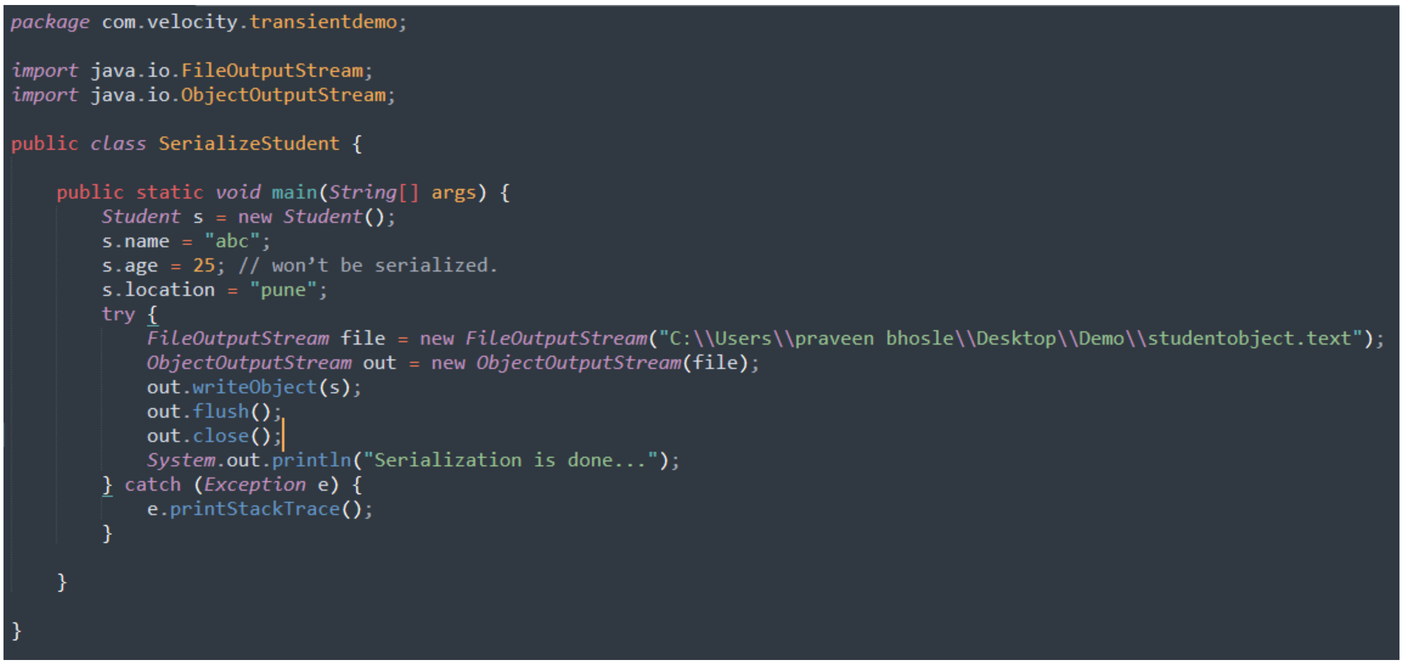
```java
package com.velocity.common;

import java.io.FileInputStream;
import java.io.ObjectInputStream;

public class DeserializeEmployee {

    public static void main(String[] args) {
        try {

            FileInputStream file = new FileInputStream("C:\\Users\\praveen bhosle\\Desktop\\Demo\\employeeobject.text");
            ObjectInputStream in = new ObjectInputStream(file);
            Object obj1 = in.readObject();
            Object obj2 = in.readObject();
            Object obj3 = in.readObject();

            Employee e1 = (Employee) obj1;
            Employee e2 = (Employee) obj2;
            Employee e3 = (Employee) obj3;

            System.out.println(e1.id + " " + e1.name);
            System.out.println(e2.id + " " + e2.name);
            System.out.println(e3.id + " " + e3.name);
            in.close();

        } catch (Exception e) {
            System.out.println(e);
        }

    }
```

- The ObjectOutputStream and ObjectInputStream are used to serialize and de-serialize objects respectively.
- If the superclass implements serializable interface, then all its subclasses will be serializable by default.
- All static members of class are not serialized because static members are related to class only, not to object.
- If we don't want to serialize some fields of class then we use the transient keyword. If any member is declared as transient then it won't be serialized.
- In case of array or collection, all the objects of array or collection must be serializable; if any object is not serializable then the serialization will fail.
- The serialization associated with each serializable class has a version number called Serial Version UID.
- It is used during de-serialization to verify that the sender and receiver of a serialized object have loaded classes for that and are compatible with respect to serialization.
- If the receiver is loaded with different version of a class that has different serial version UIDs than the corresponding sender's class, then de-serialization will result in an invalid Class Exception.
- A Serializable class can declare its own serial version UID explicitly by declaring a field named serial version UID that must be static, final and of type long.
- If a superclass variable is made transient, then after de-serialization, it gives default value like zero or null.

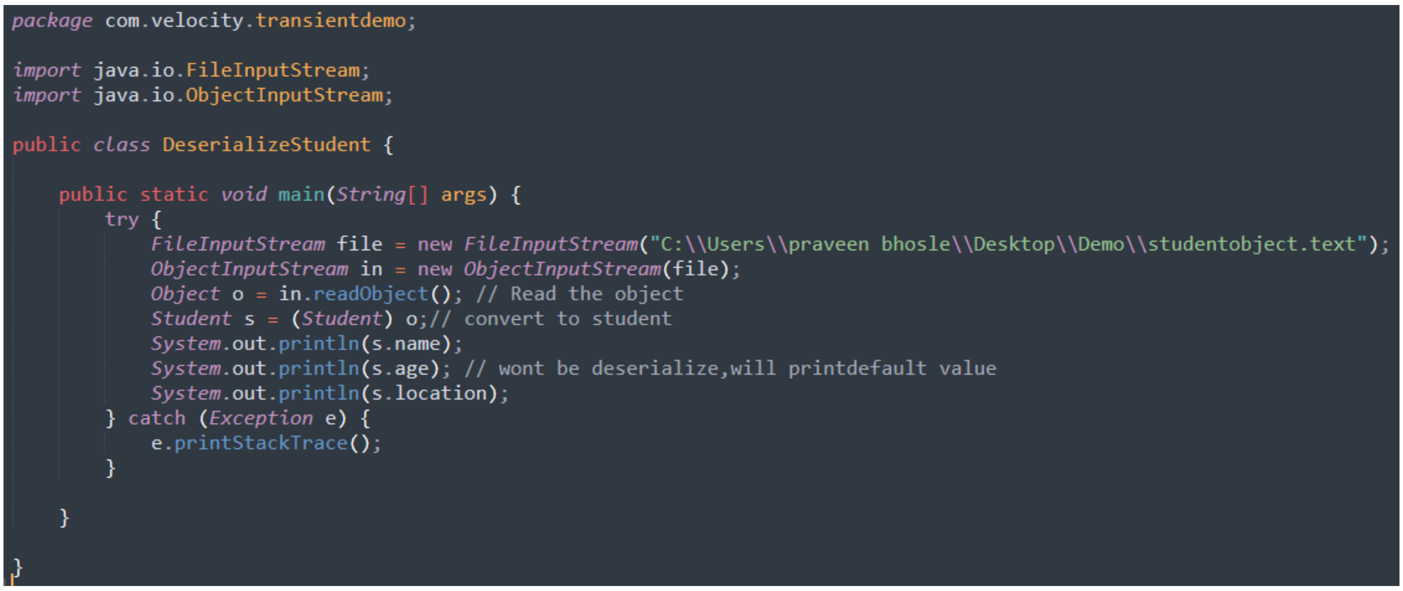
Example 2



```java
package com.velocity.transientdemo;

import java.io.Serializable;

public class Student implements Serializable {

    private static final long serialVersionUID = -570075030785935779L;
    String name;
    transient int age;
    String location;

}
```



```java
package com.velocity.transientdemo;

import java.io.FileOutputStream;
import java.io.ObjectOutputStream;

public class SerializeStudent {

    public static void main(String[] args) {
        Student s = new Student();
        s.name = "abc";
        s.age = 25; // won't be serialized.
        s.location = "pune";
        try {
            FileOutputStream file = new FileOutputStream("C:\\Users\\praveen bhosle\\Desktop\\Demo\\studentobject.text");
            ObjectOutputStream out = new ObjectOutputStream(file);
            out.writeObject(s);
            out.flush();
            out.close();
            System.out.println("Serialization is done...");
        } catch (Exception e) {
            e.printStackTrace();
        }

    }

}
```



```java
package com.velocity.transientdemo;

import java.io.FileInputStream;
import java.io.ObjectInputStream;

public class DeserializeStudent {

    public static void main(String[] args) {
        try {
            FileInputStream file = new FileInputStream("C:\\Users\\praveen bhosle\\Desktop\\Demo\\studentobject.text");
            ObjectInputStream in = new ObjectInputStream(file);
            Object o = in.readObject(); // Read the object
            Student s = (Student) o;// convert to student
            System.out.println(s.name);
            System.out.println(s.age); // wont be deserialize,will printdefault value
            System.out.println(s.location);
        } catch (Exception e) {
            e.printStackTrace();
        }

    }

}
```