THREAD LIFE CYCLE IN JAVA

A thread can exist in different stages. There are 5 states to be specific. This is taken care by the JVM. The states are mentioned below:

1. New: It is used to create an instance of Thread before the start method is called.
2. Runnable: The thread is eligible to be run after calling start but it is not scheduled.
3. Running: The thread runs if it is scheduled to do so.
4. Non-Runnable: In this state the thread is not runnable but is functioning.
5. Terminated: The thread has reached the end so it's not alive or terminated.

SERIALIZATION

It is a mechanism using Java to write the state of an object into byte stream. This is done using the writeObject() method of ObjectOutputStream class.

The serializable interface is implemented on the class whose objects are to be carried on. When a class implements serializable its objects can be converted into stream.

ObjectOutputStream Class: It is used to write Java objects and primitive data types to an OutputStream.

ObjectInputStream Class: It is used to deserialize objects and primitive data using ObjectOutputStream.

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;

public class Test2 {
    public static void main(String[] args) throws Exception{
        ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(new File("employee.txt")));

        Employee emp = new Employee();
        emp.setId(100);
        emp.setName("java");
        oos.writeObject(emp);
        oos.close();
        System.out.println("completed");

        ObjectInputStream ois = new ObjectInputStream(new FileInputStream(new File("employee.txt")));
        Employee obj = (Employee) ois.readObject();
        System.out.println(obj.getName()+"--"+obj.getId());
//        System.out.println(obj.equals(emp));
        ois.close();

    }
}
```

```
class Employee implements Serializable{

    private static final long serialVersionUID = 8885206033089891707L;
    private int id;
    private String name;

    public Employee() {
    }

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

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

In this example it can be observed that the class employee implements Serializable interface. It has variables id and name who access specifiers are private so we use setters and getter to communicate. And we use super class and to improve reusability ‘this’ keyword is used. A random serial number is generated by the IDE as shown.

In the main function we create a FileOutputStream and pass a new file. This again is passed a parameter to the ObjectOutputStream. Then we create an instance of Class Employee emp. Later add the values of Id and Name using setters. The created objected ‘oos’ now has method writeObject() to which emp is passed. We close the ‘oos’ by using close () method.

Similarly to retrieve the input we use the ObjectInputStream and follow the same steps by replacing FileInputStream. Now a new instance obj of Class Employee is created. The instance of ObjectInputStream ‘ois’ has a method readObject() which is used to read the object and this is type cast with Employee Class. In the end we print the values using getters and then close the ‘ois’ using close() method.