Object Serialization

Agenda

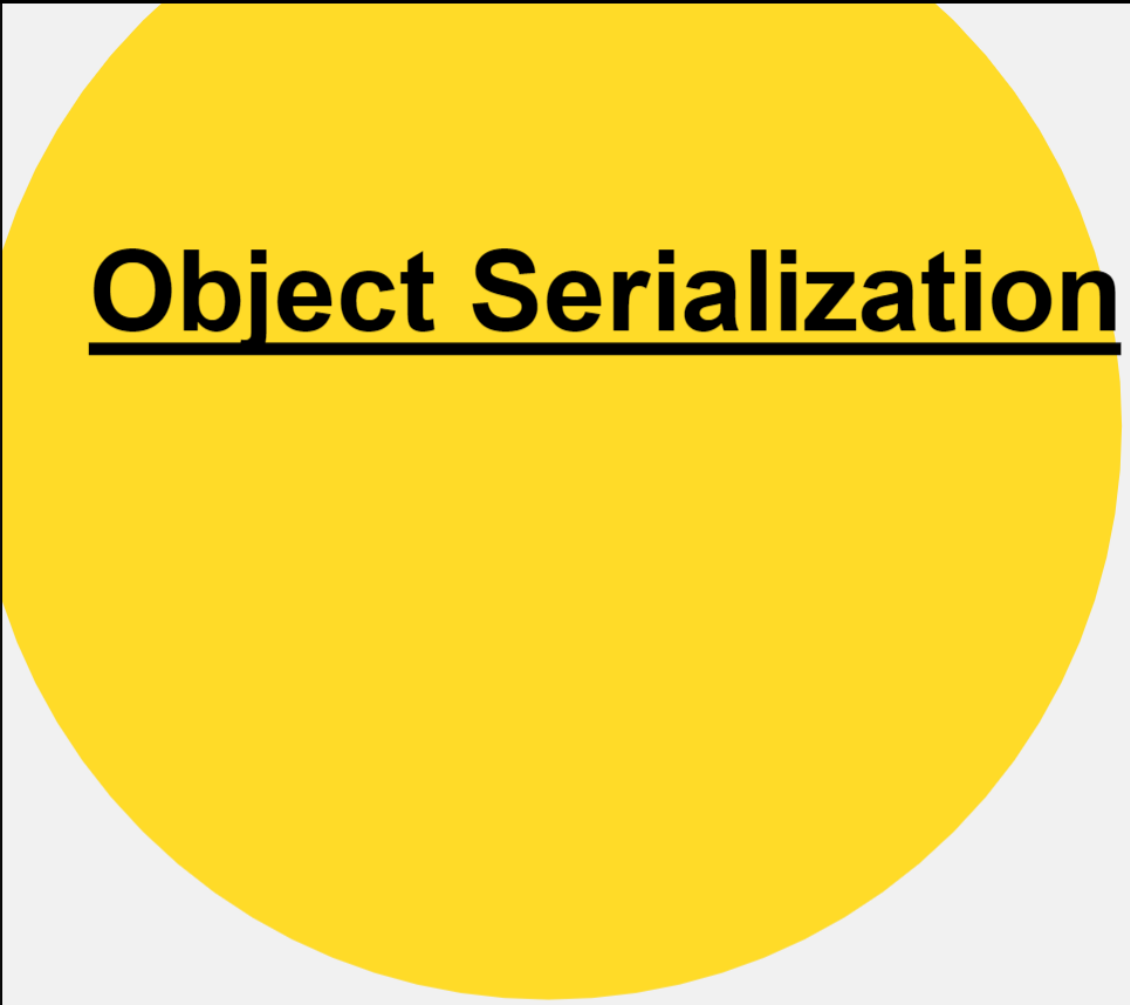
1 **Object Serialization**

Objectives

At the end of this module, you will be able to:

- Understand Object Serialization

Object Serialization



Serialization

- Object serialization is the process of saving as object's state to a sequence of bytes (on disk), as well as the process of rebuilding those bytes into a live object at some future time
- The Java Serialization API provides a standard mechanism to handle object serialization

- You can only serialize the objects of a class that implements Serializable interface

Serializing Objects

- **How to Write to an ObjectOutputStream**

```
FileOutputStream out = new FileOutputStream("theTime");
ObjectOutputStream s = new ObjectOutputStream(out);
s.writeObject("Today");
s.writeObject(new Date());
s.flush();
```

- **How to Read from an ObjectOutputStream**

```
FileInputStream in = new FileInputStream("theTime");
ObjectInputStream s = new ObjectInputStream(in);
String today = (String)s.readObject();
Date date = (Date)s.readObject();
```

Object Serialization

```
package m10.io;
import java.io.*;

public class MyClass implements Serializable {
  String s;
  int i;
  double d;
  public MyClass(String s, int i, double d) {
       this.s = s;
       this.i = i;
       this.d = d;
  }
  public String toString() {
       return "s=" + s + "; i=" + i + "; d=" + d;
  }
}
```

Object Serialization (Contd.).

```
public class SerializationDemo {
  public static void main(String args[]) {
        try {
              MyClass object1 = new MyClass("Hello", -7, 2.7e10);
                 System.out.println("object1; " + object1);
                 FileOutputStream fos = new FileOutputStream("serial");
              ObjectOutputStream oos = new ObjectOutputStream(fos);
              oos.writeObject(object1);
              oos.flush();
              oos.close();
        }
        catch(Exception e) {
            System.out.println("Exception during serialization:"+ e);
              System.exit(0);
        }
```

Object Serialization (Contd.).

```
// Object Deserialization
        try {
                MyClass object2;
                FileInputStream fis = new FileInputStream("serial");
                ObjectInputStream ois = new ObjectInputSream(fis);
                object2 = (MyClass)ois.readObject();
                ois.close();
                System.out.println("object2: " + object2);
        }
        catch(Exception e) {

                System.out.println("Exception during deserialization: "
   + e);
                System.exit(0);
        }
  }
}
```

The keyword : transient

- *transient* keyword is used in Object Serialization.
- By default, when you serialize an object, all its fields are serialized except for static variables.
- When you construct this object back from its persistent state, you will get the values of all the fields that are serialized(except static variables)
- If you do not want to store the value of a particular non static field, then you can declare this field as transient.
- This keyword is used only with a variable declaration.

Object Serialization (Contd.).

```
// Object Deserialization
        try {
                MyClass object2;
                FileInputStream fis = new FileInputStream("serial");
                ObjectInputStream ois = new ObjectInputSream(fis);
                object2 = (MyClass)ois.readObject();
                ois.close();
                System.out.println("object2: " + object2);
        }
        catch(Exception e) {

                System.out.println("Exception during deserialization: "
   + e);
                System.exit(0);
        }
  }
}
```

Demo : transient

```
import java.io.*;
class Xyz implements Serializable {
    double d1;
    transient double d2;
    static double d3;
    void  m1() {


  System.out.println("The value of d1 is :" +d1);
  System.out.println("The value of d2 is :" +d2);
  System.out.println("The value of d3 is :" +d3);
    }
}
```

Try this demo first by declaring the variable d2 as non-transient(delete the key word transient). Try again by declaring the variable d2 as transient and observe the difference

Demo : transient(Contd.).

```
class TransientExample1 {
    public static void main(String [] args) throws IOException
  {
        Xyz x = new Xyz();
        x.d1=10.3;
        x.d2=20.5;
        x.d3=99.99;
        x.m1();
        FileOutputStream fx = new FileOutputStream("A1.xyz");
        ObjectOutputStream ox = new ObjectOutputStream(fx);
        ox.writeObject(x);
        ox.flush();
    }
}
```

Demo : transient(Contd.).

```
import java.io.*;
class TransientExample2 {
	public static void main(String [] args) {
		try {

	FileInputStream fx = new FileInputStream("A1.xyz");

	ObjectInputStream ox = new ObjectInputStream(fx);
		Xyz x = (Xyz) ox.readObject();
		x.m1();
		}
		catch(Exception e) {
		System.out.println(e);
		}
	}
}
```

Demo : transient(Contd.).

Scenario 1 : **When d2 is not transient !**

- When you compile all the three source files viz. Xyz.java, TransientExample1.java and TransientExample2.java and execute first TransientExample1 and then TransientExample2, you will get the following output (from executing TransientExample2):
- The value of d1 is :10.3
- The value of d2 is :20.5
- The value of d3 is :0.0

- In the above result, d3 is not serialized since d3 is declared as static.

Demo : transient(Contd.).

Scenario 1 : **When d2 is transient !**

- After declaring d2 as transient, when you compile Xyz.java and then execute first Transient Example1 and then TransientExample2, you will get the following output :
- The value of d1 is :10.3
- The value of d2 is :0.0
- The value of d3 is :0.0

- In the above result, d2 is not serialized since it is declare as transient.

Thank You