**serialVersionUID**

* **serialVersionUID:**

In serialization both sender and receiver need not be same person, need not to use same machine and need not be from the same location.

The persons may be different, the machines may be different and locations may be different.

In serialization both sender and receiver should have .class file at the beginning only just state of object is travelling from sender to receiver.

At the time of serialization with every object sender side JVM will save a unique identifier.

JVM is responsible to generate this unique identifier based on .class file.

At the time of deserialization receiver side JVM will compare unique identifier associated with object with local class unique identifier. If both are matched then only deserialization will be performed otherwise we will get runtime exception saying:

InvalidClassException

This unique identifier is nothing but serialVersionUID.

* **Problems of depending on default serialversion UID generated by JVM:**

1. Both sender and receiver should use same JVM with respect to vendor, platform and version. Otherwise receiver unable to deserialize because of different serialVersionUID’s.
2. Both sender and receiver should use same .class file version. After serialization if there is any change in .class file at receiver side then receiver will be unable to deserialize.
3. To generate serialVersionUID internally JVM may use complex algorithm which may create performance problems.

We can solve/overcome above problems by configuring our own serialVersionUID.

We can configure our own serialVersionUID as follows.

private static final long serialVersionUID = 1l;

* **Example:**

class Dog1{

private static final ling serialVersionUID = 1l;

int i = 10;

int j = 20;

}

class Sender{

public static void main(String[] args) throws Exception{

Dog1 d1 = new Dog1();

FileOutputStream fos = new FileOutputStream(“abc.ser”);

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(d1);

}

}

class Receiver{

public static void main(String[] args) throws Exception{

FileInputStream fis = new FileInputStream(“abc.ser”);

ObjectInputStream ois = new ObjectInputStream(fis);

Dog1 d2 = (Dog) ois.readObject();

System.out.println(d2.i+”…”+d2.j);

}

}

Note:

In the above program after serialization if we perform any change to the .class file at receiver side we won’t get any problem at the time of deserialization.

In this case sender and receiver not required to maintain same JVM versions.

Some IDE’s prompt programmer to enter serialVersionUID explicitly.

Some IDE’s may generate serialVersionUID automatically.