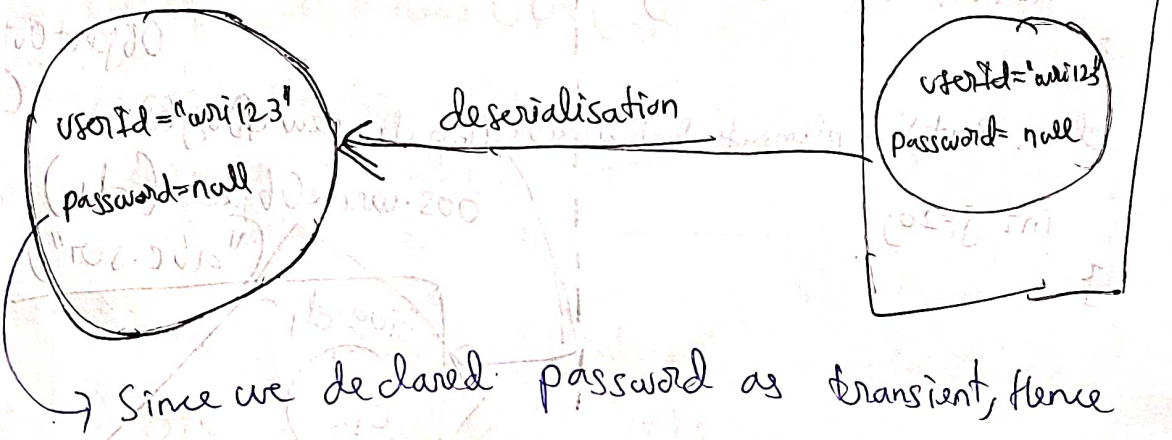
(Part-6)

Why we need Customised Serialisation?

→ (To avoid Data Loss)

```
Class Account implements Serializable {
String userId = "uri 123"
transient String password = "#132 go"
}
```



→ Since we declared password as transient, Hence it did not participate in serialisation, and there was loss data during serialisation & deserialisation.

To transfer this password also some additional work is required to be done at Sender Side as well as Receiver Side as well.

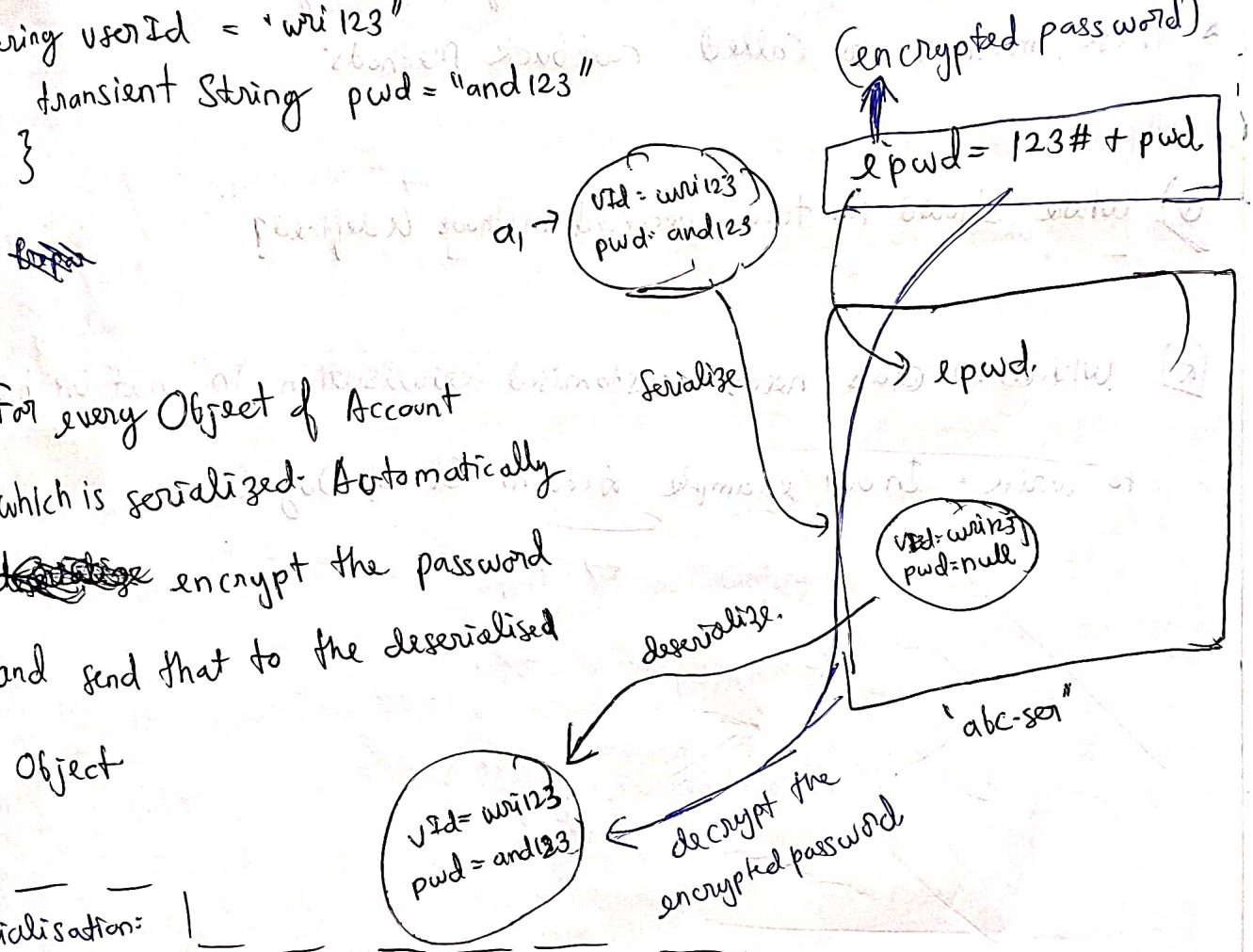
→ (this additional work is called Customised Serialisation)

(Part 7)

[How to implement Customised Serialisation]

Lets say that extra work done by sender and done by receiver is encryption of password and decryption of password.

```
class Account implements Serializable{
String userId = "uri123"
   transient String pwd = "and123"
}
```



For every Object of Account which is serialized Automatically encrypt the password and send that to the deserialised Object

Serialisation: Prepare Encrypted Password and write it to file.

Deserialisation: Read Encrypted Password from file and set value in the Object

⇒ this is called customised Serialisation

two methods which we use to implement customised Serialisation.

this needs to be used Serialisation.

```
private void writeObject(ObjectOutputStream oos) throws Exception
```

this needs to be used during Deserialisation.

```
private void readObject(ObjectInputStream ois) throws Exception.
```

* These two methods will be automatically called by JVM.
* these methods are called callback Methods.

Q) Where should the two specified methods be defined?

A) Whichever class needs customised Serialisation in that we have to write. In our example Account Class, Dog Class.

(Part 8)

(Coding of Customised Serialisation)

```
class Account implements Serializable{
    String username = "wri123";
    transient String pwd = "#123wb";

    private void writeObject(OOS os) throws Exception
    {
        os.defaultWriteObject();
        String epwd = "123" + pwd;
        os.writeObject(epwd);
    }

    private void readObject(OIS is) throws Exception
    {
        is.defaultReadObject();
        String epwd = (String) is.readObject();
        pwd = epwd.subString(3);
    }
}
```

(***) default serialisation (password is null)

(writing encrypted password to file)

(***) default deserialisation pwd is null

Read encrypted password from file and then decrypt it.

```
class CustomSerializeDemo{
    psvm(){
        Account a1 = new Account();
        FOS fos = new FOS("abc.ser");
        OOS oos = new OOS(fos);
        oos.writeObject(a1);

        FIS fis = new FIS("abc.ser");
        OIS ois = new OIS(fis);
        Account a2 = ois.readObject();
```

(this method is called)

(this method is called)

(Part-9)

Lets say there are more than one transient variables, and you want to use customised serialisation, in that case how you can do?

As we remember the order in which we serialize in the same order we have to deserialize. Its the same in customised serialisation.

```
Class Account implements Serializable{

    String id = "sri123"

    Transient String pwd = "#123@";   → (2 transient variables)
    Transient int pin = 1234;

    private void writeObject(ObjectOutputStream oos) throws Exception.
    {
        oos.writeObject();  → default serialisation.
        String epwd = "123" + pwd;
        int epin = 4444 + pin;
        oos.writeObject(epwd);
        oos.writeInt(epin);
    }

    private void readObject(ObjectInputStream ois) throws Exception {
        ois.readObject();  → default deserialisation.
        String epwd = (String) ois.readObject();
        int epin = ois.readObject();
        pwd = epwd.substring(3);
        pin = epin - 4444;
    }.
```