THE DECORATOR PATTERN

THE DECORATOR PATTERN

ALL INPUT STREAMS DERIVE FROM A COMMON ABSTRACT CLASS InputStream

WHICH CONTAINS STANDARD OPERATIONS SHARED BY ALL STREAMS

- CLOSE()
- READ()
- RESET()

THEN, FOR EACH TYPE OF INPUT STREAM (I.E. EACH INDEPENDENT VARIABLE), THERE IS A SEPARATE CLASS DERIVING FROM INPUTSTREAM

FileInputStream ObjectInputStream AudioInputStream

GZIPInputStream ByteArrayInputStream

NOW HERE IS THE CRUCIAL, SUPER-IMPORTANT BIT: EVERY ONE OF THESE OBJECTS CAN BE CONSTRUCTED FROM AN OBJECT OF TYPE InputStream

INPUTSTREAM OBJECTS
CAN BE CHAINED TO GET
YET MORE INPUTSTREAM
OBJECTS

THIS CHAINING OF OBJECTS - ALL OF WHICH
DESCEND FROM THE SAME INTERFACE OR
ABSTRACT BASE CLASS -

TO GET MORE OBJECTS OF THE
SAME BASE CLASS -

IS THE HALLMARK OF THE

DECORATOR PATTERN

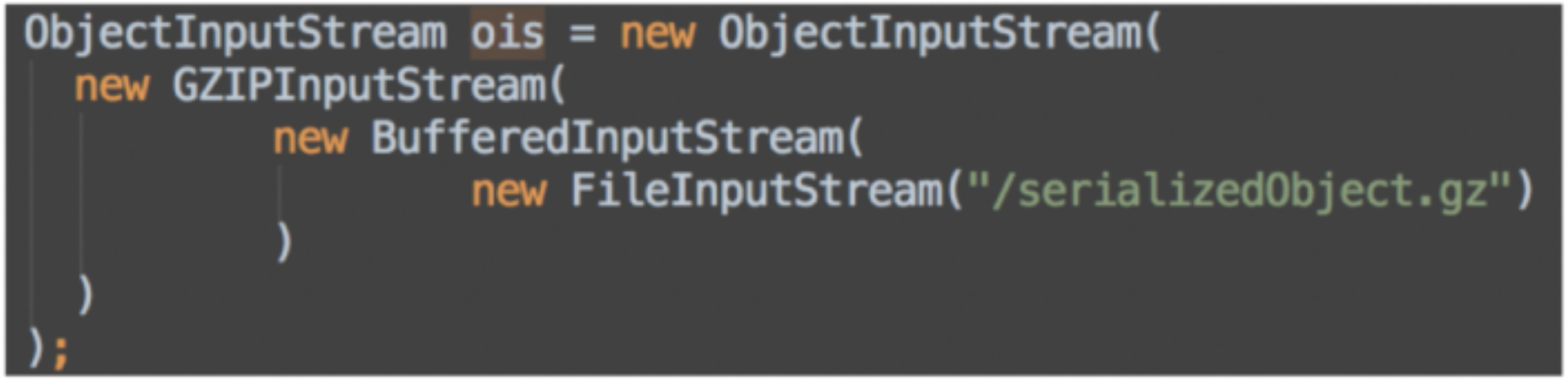
FOR INSTANCE - CONSIDER THAT WE WROTE SOME CODE THAT SAVES OBJECTS TO A FILE, AND THEN ZIPS THAT FILE

SAVING OBJECTS TO FILES IS A STANDARD OPERATION CALLED

SERIALIZATION

NOW, WHEN TIME COMES TO USE THESE OBJECTS - WE NEED TO UNZIP THE FILE, READ IN THE OBJECTS AND THEN USE THEM

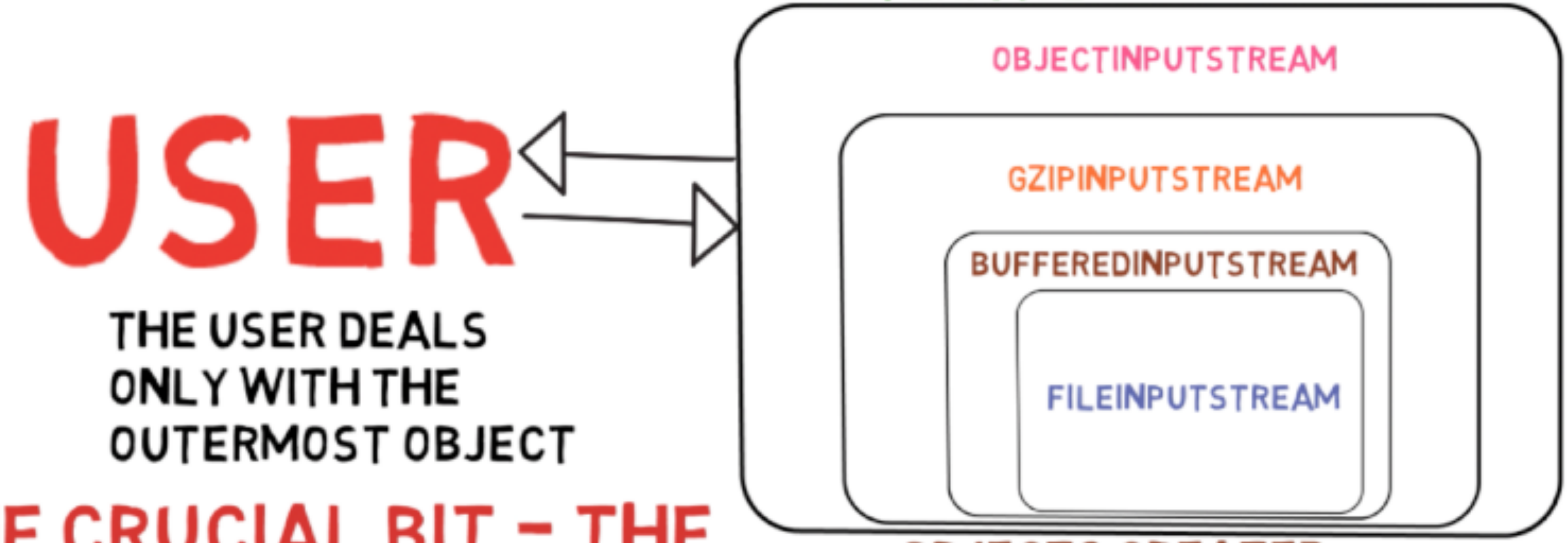
WE CREATE A CHAIN OF INPUTSTREAM OBJECTS



```
ObjectInputStream ois = new ObjectInputStream(
  new GZIPInputStream(
          new BufferedInputStream(
                  new FileInputStream("/serializedObject.gz")
          )
  )
);
```

THIS DIAGRAM SUMS UP THE DECORATOR PATTERN

A BUNCH OF OBJECTS, EACH IMPLEMENTING THE INPUTSTREAM INTERFACE



THE USER DEALS ONLY WITH THE OUTERMOST OBJECT

OBJECTS CREATED IN A CHAIN, FROM 1PRECEDING OBJECT

THE CRUCIAL BIT - THE CHAINED OBJECTS ARE INDEPENDENT OF ONE ANOTHER