JAVA PROGRAMMING BASICS

Module 3: Java Standard Edition

Training program

1. Java I/O Streams
2. Java Serialization
3. Java Database Connectivity
4. Java GUI Programming
5. The basics of Java class loaders
6. Reflections
7. Annotations
8. The proxy classes
9. Java Software Development
10. Garbage Collection

Module contents

1. Java Serialization
   - Java - Serialization
   - Serializing an Object
   - The transient modifier
   - Complex Objects Serialization
   - Serialization with Inheritance
   - Custom Serialization in Java
   - Java Externalizable Interface
   - XML & JSON serialization

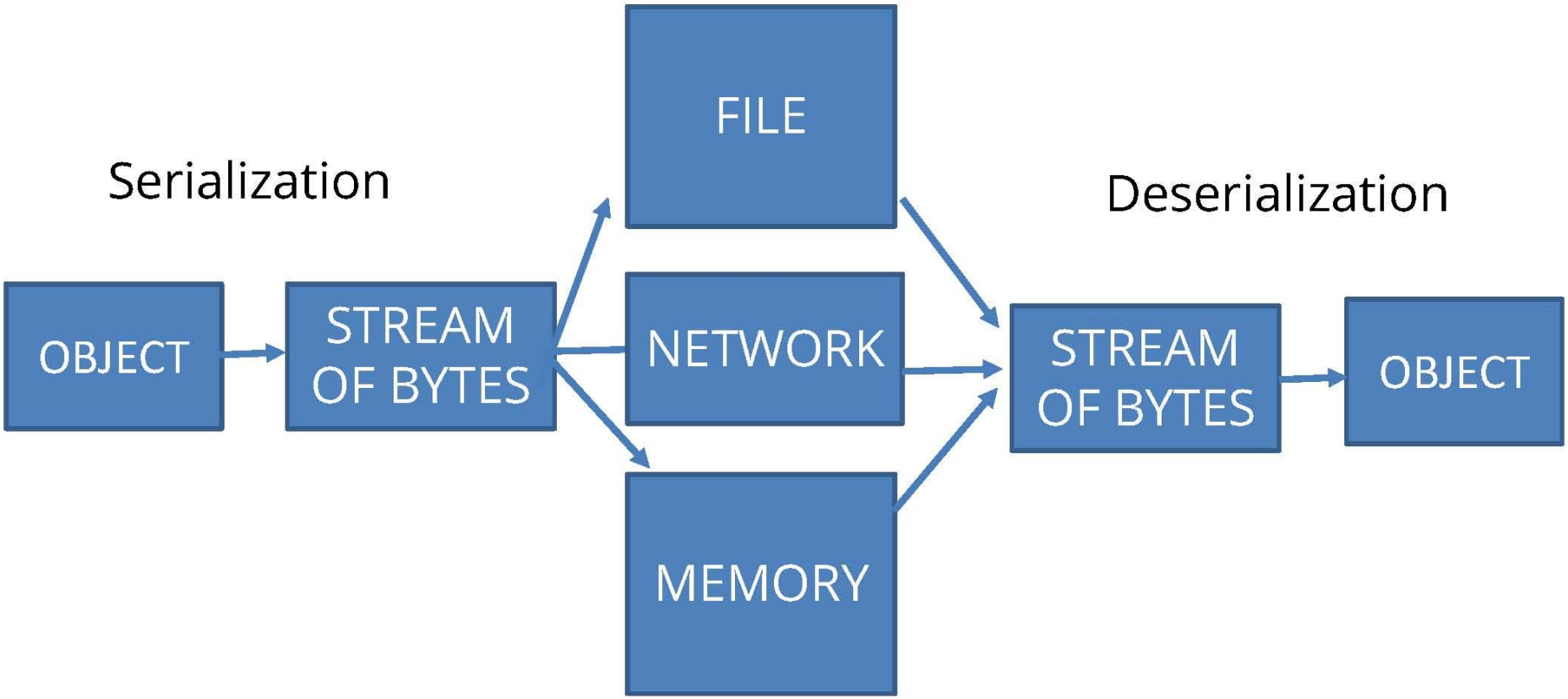
Module contents

1. Java Serialization
   - Java - Serialization
   - Serializing an Object
   - The transient modifier
   - Complex Objects Serialization
   - Serialization with Inheritance
   - Custom Serialization in Java
   - Java Externalizable Interface
   - XML & JSON serialization

Java Serialization 1/9

- Often, when a program is running, it is necessary to save the state of its objects (intermediate and final). **Serialization** and **deserialization** are Java technologies that store objects as a sequence of bytes and restore objects from such a sequence.
- A serialized object as a sequence of bytes can be written to a file, transmitted over the network, written to a buffer in memory. And this sequence can be deserialized into an object by this or another program on the current or another computer, regardless of the operating system running on it.
- The following information is included when an object is serialize:
  - the class information needed to reconstruct the object;
  - the values of all serializable non-transient and non-static members, including those that are inherited.
- Object methods and constructors are not saved as part of the serialized IO stream, since the receiving side must have an object class to interpret the serialized byte stream.

Java Serialization 2/9



Module contents

1. Java Serialization
   - Java - Serialization
   - Serializing an Object
   - The transient modifier
   - Complex Objects Serialization
   - Serialization with Inheritance
   - Custom Serialization in Java
   - Java Externalizable Interface

Java Serialization

- Uses **java.io.Serializable** as a marker interface
- Consists of **java.io.ObjectInputStream/ObjectOutputStream** with related classes and interfaces
- Has "**Java Object Serialization Specification**" that documents both API and object serialization stream protocol
- Is part of any Java platform (even Android has it)
- If class not implements Serializable interface, it serialization throws java.io.NotSerializableException

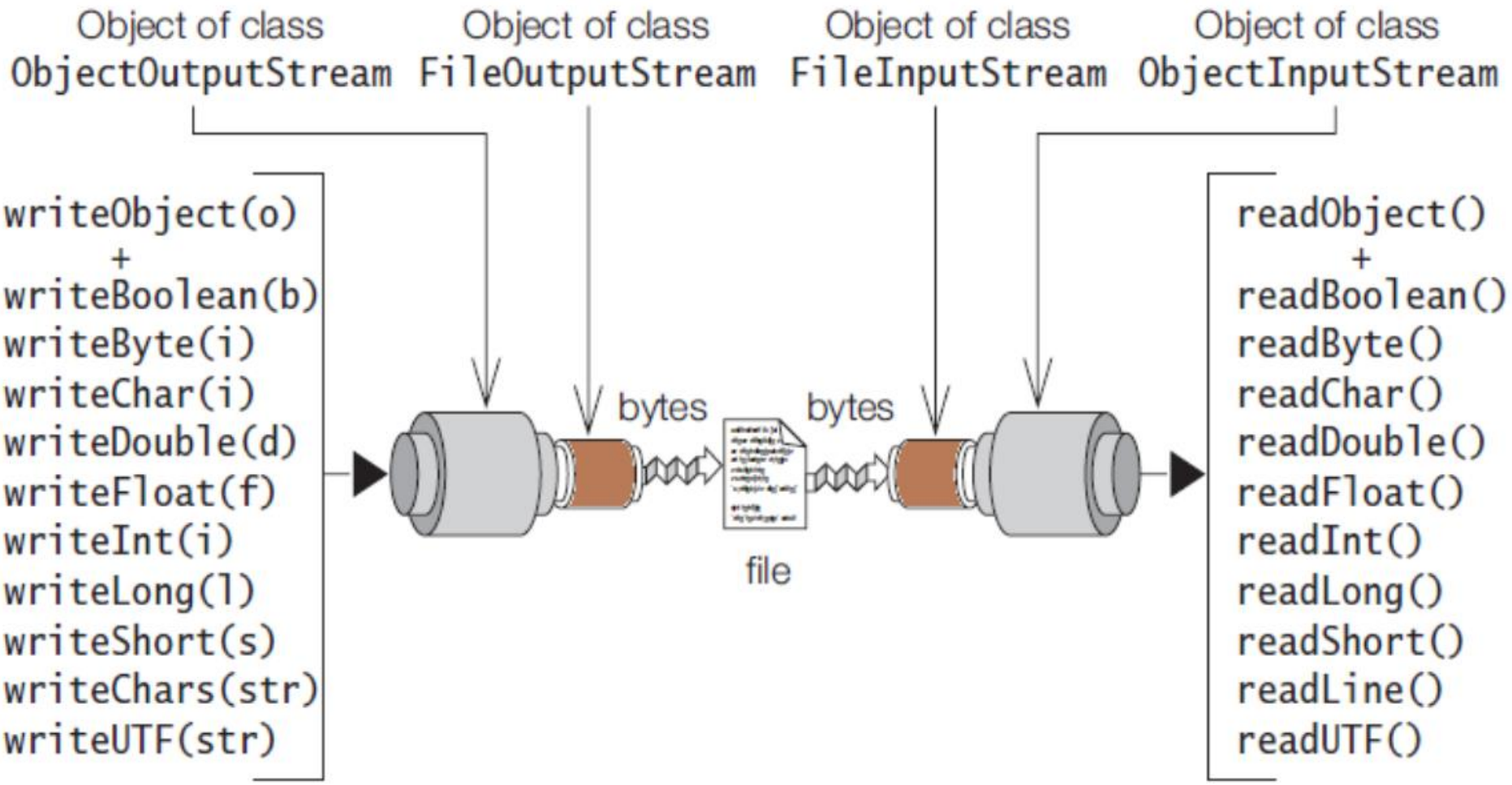
```
public interface Serializable {}
```

Java Serialization

- An **ObjectOutputStream** is used to save the object state to file (to serialize object).
- An ObjectOutputStream must wrap (chain) an OutputStream subclass, for example FileOutputStream.
- We can use for this purpose ObjectOutputStream method:

  public final void **writeObject(Object obj)** throws IOException
- An **ObjectInputStream** is used to restore the object state from file (to deserialize object).
- An ObjectInputStream must wrap (chain) an InputStream subclass, for example FileInputStream.
- We can use for this purpose ObjectInputStream method:
  public final **Object readObject()** throws IOException, ClassNotFoundException

Serialization-deserialization Stream Chaining



For serialization-deserialization to/from file

Java Serialization - serialVersionUID

- During serialization, an unique constant *serialVersionUID*
- (of type long) is calculated as hash-function from the following members of the serializing class :
  - field names and their modifiers (field values are not taken into account);
  - constructors and methods declarations, including signatures, return types and modifiers (exception declarations are ignored). Constructor and method bodies are not taken into account.
- The ***serialVersionUID*** value is placed in the serialization file.
- While deserialization, the same value is obtained for the restored object from class.
- If the numbers do not match, a java.io.InvalidClassException will be thrown.

See serialization\basic\Student

Java Serialization - serialVersionUID

- But if You add a private static final long *serialVersionUID* field to the class and set it to an arbitrary value (the standard value of 1L is often used), then the *serialVersionUID* calculation mechanism will be disabled and the *serialVersionUID* value we specified will be written to the file and used for deserialization.
- With such fixed *serialVersionUID* value You can:
  - add/remove class fields;
  - move fields around the class definition, etc.
- Since JDK 14 @Serial annotation was added. **It has not influence on serialization mechanism**. It only checks the correct field and method used by serialization signature (like @Override annotation).
  But it need compiler options:
  javac -Xlint:serial Main.java
  for warning printing.

See serialization\basic\Student

Java Serialization - serialVersionUID

- You can generate the *serialVersionUID* with the **serialver** tool:

```
serialver -classpath bin basic.Student
basic.Student:    private static final long
serialVersionUID = 1122474873839213 4L;
```

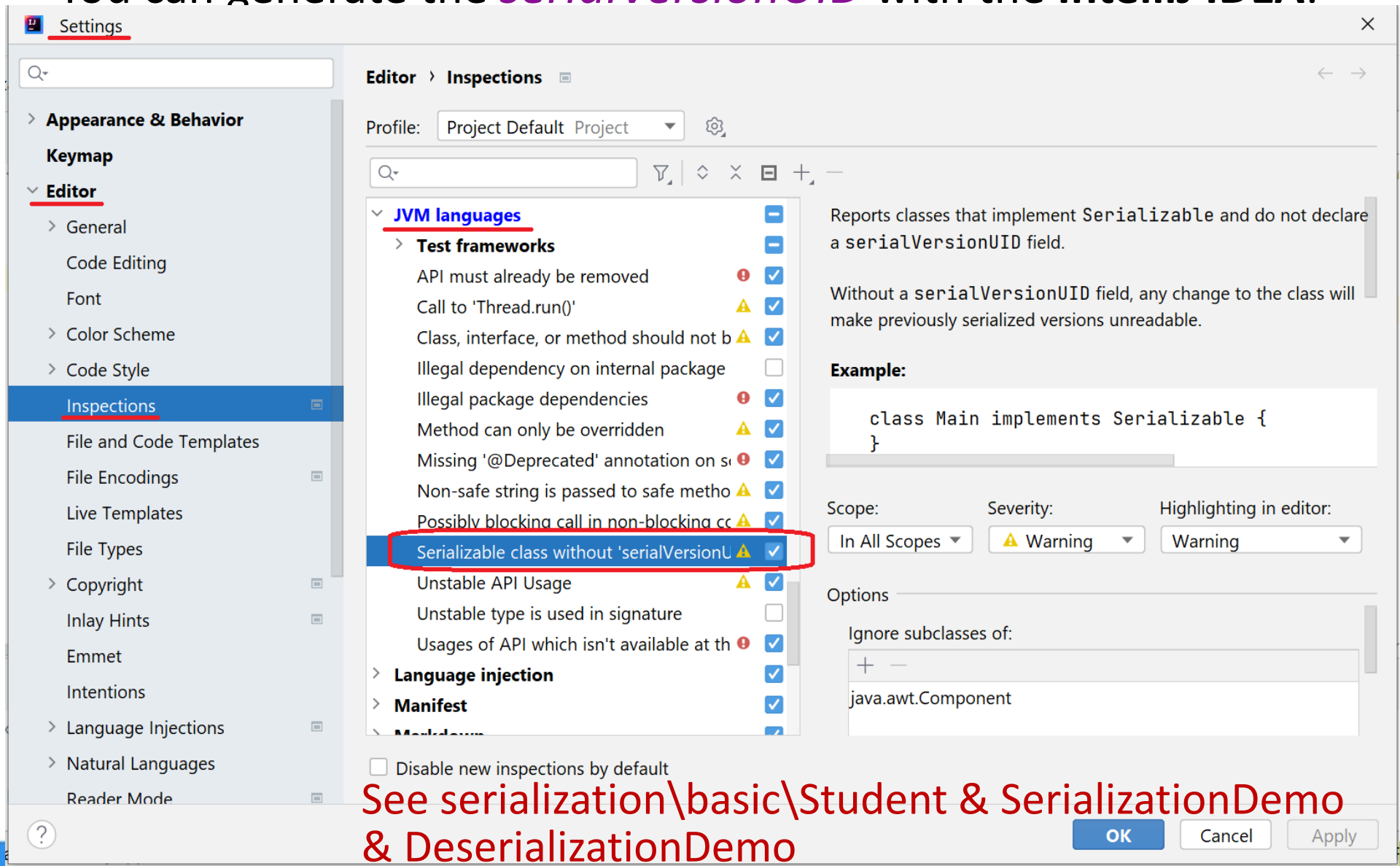
```
c:\Users\kgp\JavaStudy>serialver -classpath out\production\JavaStudy lesson38.serialization.basic.Student
lesson38.serialization.basic.Student:    private static final long serialVersionUID = 11224748738392134L;

c:\Users\kgp\JavaStudy>
```

See serialization\basic\Student

Java Serialization - *serialVersionUID*

- You can generate the *serialVersionUID* with the **IntelliJ IDEA**:



See serialization\basic\Student & SerializationDemo & DeserializationDemo

Java Serialization explore

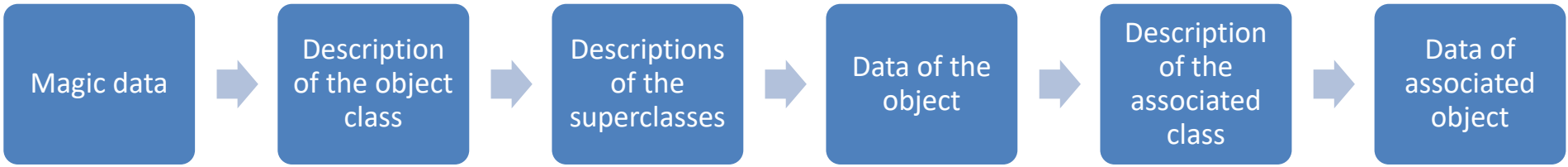
For **SerializationStudyApp** project **basic** package:

- Add and remove **implements Serializable** for class **Student** and run **SerializationDemo** program
- Comment **private static final long serialVersionUID = 1L** field in **Student** class and add **private String patronymicName** field and run **DeserializationDemo** program
- In console run
  serialver -classpath bin basic.Student
  and check if serialVersionUID changed while:
  adding field, adding field value,
  changing constructor modifier,
  adding throws clause to constructor and method,
  changing body of constructor and method.

Java Serialization

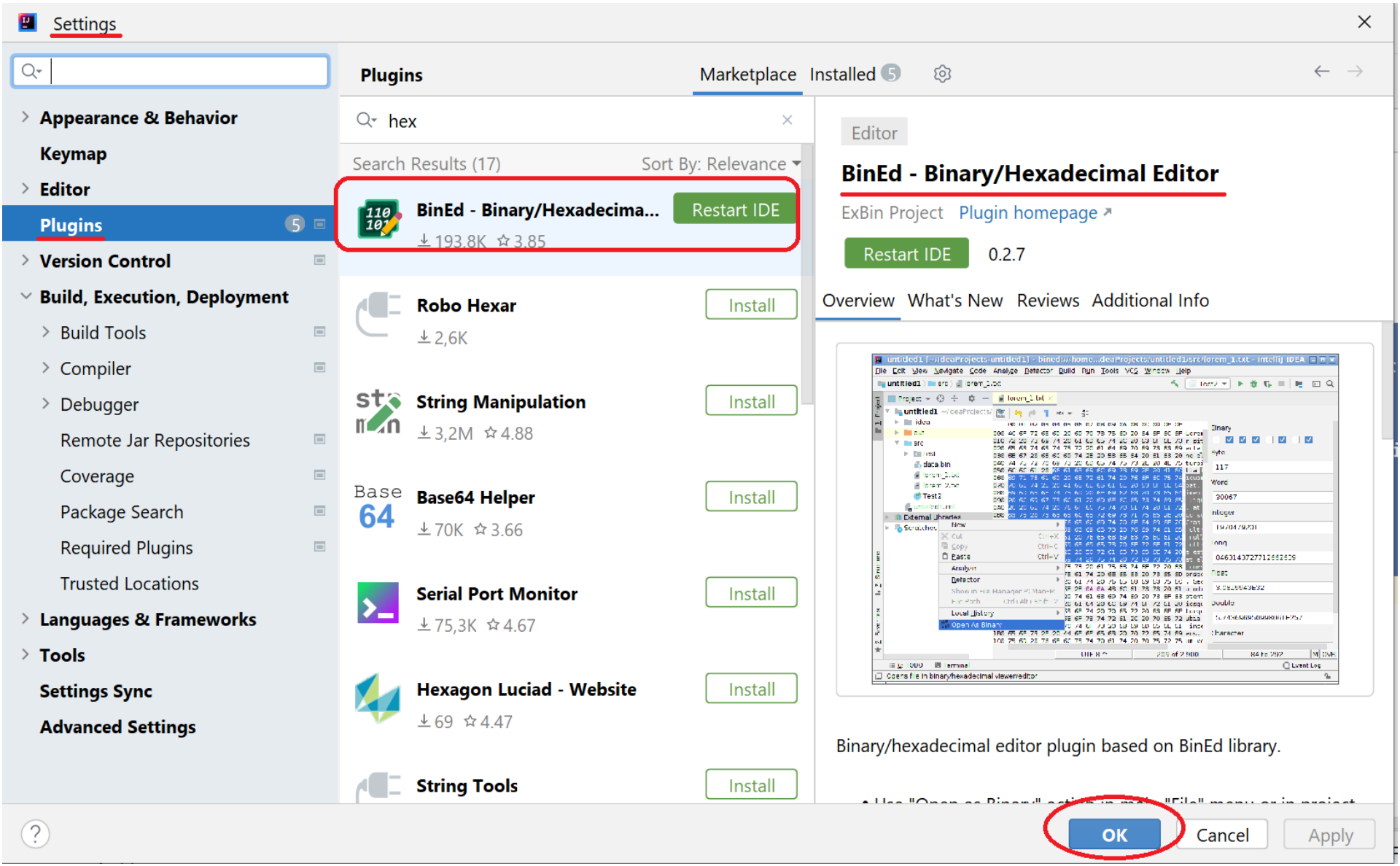
The serialization algorithm does the following things:

1) recording metadata about the class associated with the object;
2) recursively writing superclass descriptions until java.lang.Object is reached;
3) after the end of the metadata recording, recording of the actual data associated with the instance, starts recording from the topmost superclass;
4) recursively writing data associated with an instance, starting with the lowest superclass

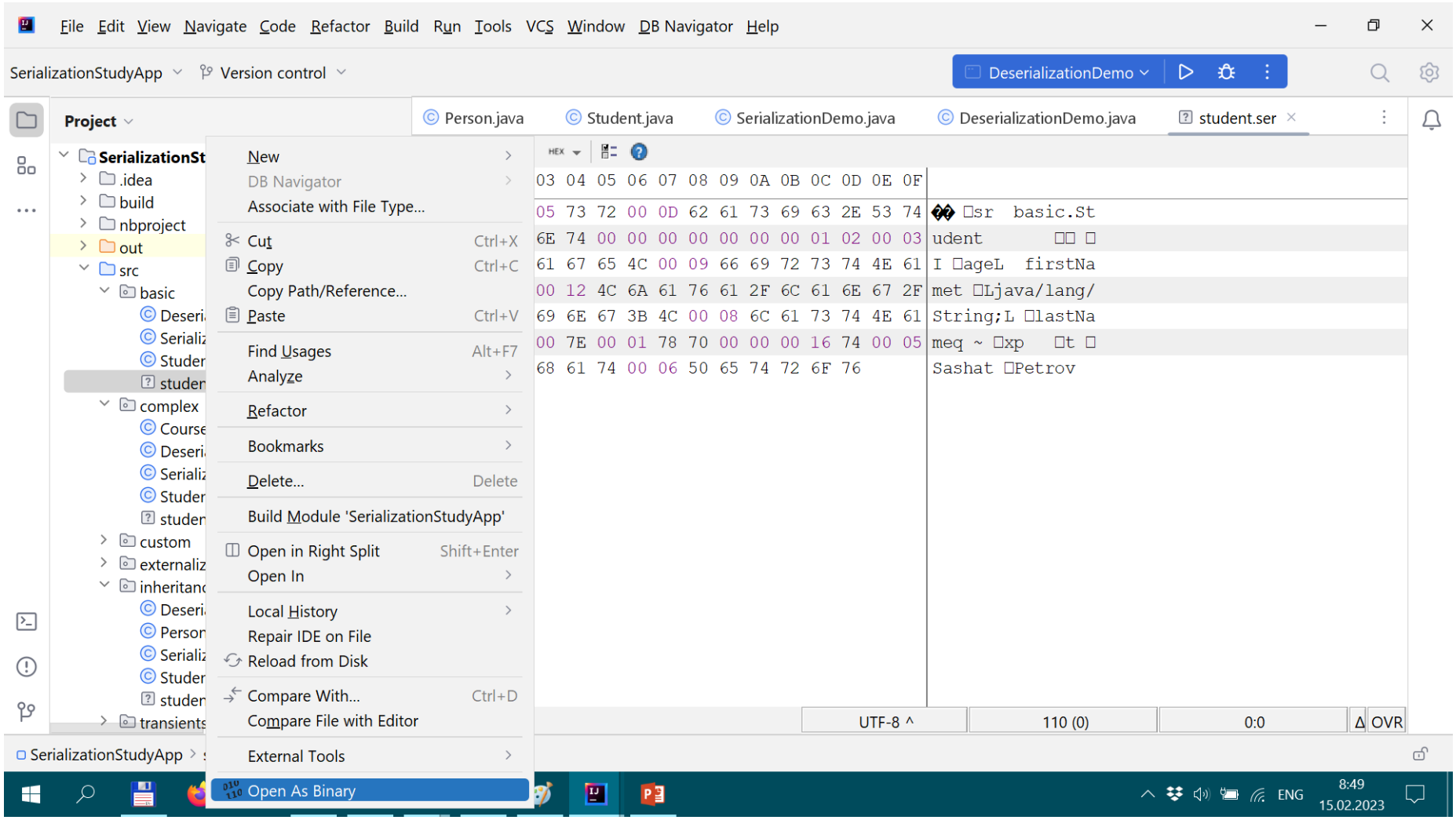


Java Object Serialization Specification - https://docs.oracle.com/en/java/javase/19/docs/specs/serialization/index.html

Serialization file format



Serialization file format



Serialization file format -1/4

student.ser

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
00000000	ac	ed	00	05	73	72	00	0d	62	61	73	69	63	2e	53	74	¬H..sr..basic.St
00000010	75	64	65	6e	74	00	00	00	00	00	00	00	01	02	00	03	udent..........
00000020	49	00	03	61	67	65	4c	00	09	66	69	72	73	74	4e	61	I..ageL..firstNa
00000030	6d	65	74	00	12	4c	6a	61	76	61	2f	6c	61	6e	67	2f	met..Ljava/lang/
00000040	53	74	72	69	6e	67	3b	4c	00	08	6c	61	73	74	4e	61	String;L..lastNa
00000050	6d	65	71	00	7e	00	01	78	70	00	00	00	16	74	00	05	meq.~..xp....t..
00000060	53	61	73	68	61	74	00	06	50	65	74	72	6f	76			Sashat..Petrov

Serialization protocol data (5 bytes):
ac ed: - STREAM_MAGIC - serialization protocol pointer
00 05: STREAM_VERSION - serialization protocol version
0x73: TC_OBJECT - a pointer that a new object has been serialized

Description of the class of the serialized object :
0x72: TC_CLASSDESC - serialized object class pointer
00 0d: class filename length from application root - 13 characters
62 61 73 69 63 2e 53 74 75 64 65 6e 74: package and class file name basic.Student
00 00 00 00 00 00 00 01: serialVersionUID = 1L
0x02: Flags value 02 means the object supports serialization.
00 03: number of fields of this class

Serialization file format -2/4

student.ser

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
00000000	ac	ed	00	05	73	72	00	0d	62	61	73	69	63	2e	53	74	¬H..sr..basic.St
00000010	75	64	65	6e	74	00	00	00	00	00	00	00	01	02	00	03	udent..........
00000020	49	00	03	61	67	65	4c	00	09	66	69	72	73	74	4e	61	I..ageL..firstNa
00000030	6d	65	74	00	12	4c	6a	61	76	61	2f	6c	61	6e	67	2f	met..Ljava/lang/
00000040	53	74	72	69	6e	67	3b	4c	00	08	6c	61	73	74	4e	61	String;L..lastNa
00000050	6d	65	71	00	7e	00	01	78	70	00	00	00	16	74	00	05	meq.~..xp....t..
00000060	53	61	73	68	61	74	00	06	50	65	74	72	6f	76			Sashat..Petrov

Description of the first field of the class:

0x49: variable type code 49 - "I", int ('B' for byte, 'C' for char, 'D' for double, 'F' for float, 'J' for long, 'L' for non-array object types, 'S' for short, 'Z' for boolean, and '[' for arrays))

00 03: variable name length

61 67 65: variable name - age

Description of the second field of the class (the type of reference variables is described in the format "field descriptor" L ClassName ;):

0x4c: start of reference variable declaration "L",

00 09: variable name length

66 69 72 73 74 4e 61 6d 65: variable name - firstName

0x74: TC_STRING - reference variable type pointer

Serialization file format -3/4

student.ser

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
00000000	ac	ed	00	05	73	72	00	0d	62	61	73	69	63	2e	53	74	¬H..sr..basic.St
00000010	75	64	65	6e	74	00	00	00	00	00	00	00	01	02	00	03	udent..........
00000020	49	00	03	61	67	65	4c	00	09	66	69	72	73	74	4e	61	I..ageL..firstNa
00000030	6d	65	74	00	12	4c	6a	61	76	61	2f	6c	61	6e	67	2f	met..Ljava/lang/
00000040	53	74	72	69	6e	67	3b	4c	00	08	6c	61	73	74	4e	61	String;L..lastNa
00000050	6d	65	71	00	7e	00	01	78	70	00	00	00	16	74	00	05	meq.~..xp....t..
00000060	53	61	73	68	61	74	00	06	50	65	74	72	6f	76			Sashat..Petrov

00 12: length of reference variable type name in the format "field descriptor"
4c 6a 61 76 61 2f 6c 61 6e 67 2f 53 74 72 69 6e 67 3b - type name Ljava/lang/String;

Description of the third class field :
0x4c: start of reference variable declaration "L",
00 08: variable name length
6c 61 73 74 4e 61 6d 65: variable name - lastName
0x71: TC_REFERENCE reference to an object already written to the stream (Ljava/lang/String;)
00 7e 00 01 ???
0x78: TC_ENDBLOCKDATA - end of description

0x70: TC_NULL - there is no parent class (otherwise it would be its description)

Serialization file format -3/4

student.ser

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
00000000	ac	ed	00	05	73	72	00	0d	62	61	73	69	63	2e	53	74	¬н..sr..basic.St
00000010	75	64	65	6e	74	00	00	00	00	00	00	00	01	02	00	03	udent..........
00000020	49	00	03	61	67	65	4c	00	09	66	69	72	73	74	4e	61	I..ageL..firstNa
00000030	6d	65	74	00	12	4c	6a	61	76	61	2f	6c	61	6e	67	2f	met..Ljava/lang/
00000040	53	74	72	69	6e	67	3b	4c	00	08	6c	61	73	74	4e	61	String;L..lastNa
00000050	6d	65	71	00	7e	00	01	78	70	00	00	00	16	74	00	05	meq.~..xp....t..
00000060	53	61	73	68	61	74	00	06	50	65	74	72	6f	76			Sashat..Petrov

Field values :
00 00 00 16: the value of a primitive type variable int (22)
0x74: TC_STRING - reference variable type pointer
00 05: длина значения ссылочной переменной в формате
53 61 73 68 61: reference variable value (Sasha)

0x74: TC_STRING - reference variable type pointer
00 06: the length of the value of the reference variable in the format
50 65 74 72 6f 76: reference variable value (Petrov)

Module contents

1. Java Serialization
   - Java - Serialization
   - Serializing an Object
   - The transient modifier
   - Complex Objects Serialization
   - Serialization with Inheritance
   - Custom Serialization in Java
   - Java Externalizable Interface

The transient modifier

- You can use **transient** modifier with field declaration to omit the serialization of the field - selective serialization.
- The **transient** modifier prevents serialization of the field, i.e. placing the field description and its value in the serialization file.
- However, during deserialization, due to casting the type of the restored object to the class of the serialized object, the name of the transient property is restored, but it has the default value.
- For **static** fields, serialization and deserialization are not performed, since these technologies apply to an object, not a class, so the value of a static field will be the same as it is set in the class or the default value if not set.

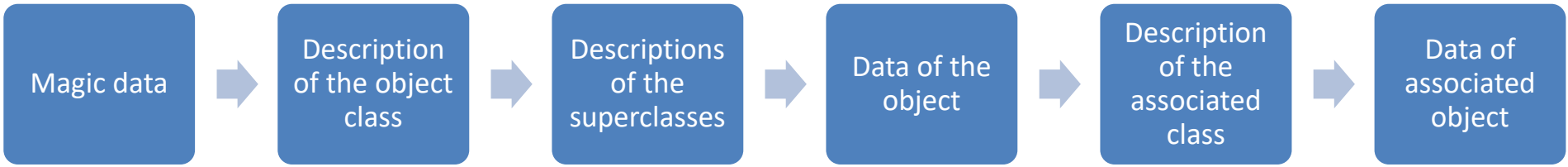
See serialization\transientstatic

Module contents

1. Java Serialization
   - Java - Serialization
   - Serializing an Object
   - The transient modifier
   - Complex Objects Serialization
   - Serialization with Inheritance
   - Custom Serialization in Java
   - Java Externalizable Interface

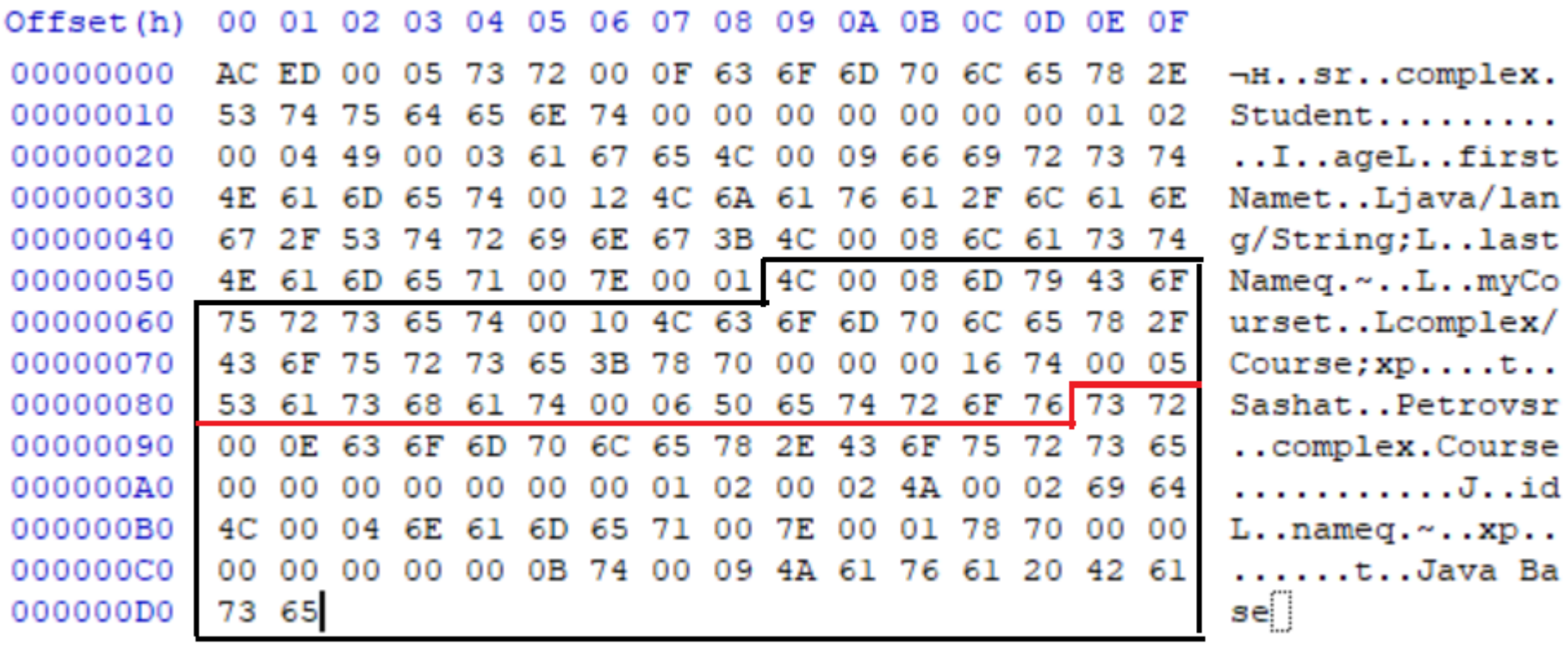
Java Serialization

- All subclasses of a serializable class are automatically serializable.
- However, if the serializable class contains a reference to an object of a non-serializable class, a **java.io.NotSerializableException** is thrown when trying to serialize.
- In other words, the resulting graph of this object is fully serializable. An object graph includes a tree or structure of object fields and its subobjects.



See serialization\complex

Serialization file format -1/4



```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   AC ED 00 05 73 72 00 0F 63 6F 6D 70 6C 65 78 2E  ¬н..sr..complex.
00000010   53 74 75 64 65 6E 74 00 00 00 00 00 00 00 01 02  Student.........
00000020   00 04 49 00 03 61 67 65 4C 00 09 66 69 72 73 74  ..I..ageL..first
00000030   4E 61 6D 65 74 00 12 4C 6A 61 76 61 2F 6C 61 6E  Namet..Ljava/lan
00000040   67 2F 53 74 72 69 6E 67 3B 4C 00 08 6C 61 73 74  g/String;L..last
00000050   4E 61 6D 65 71 00 7E 00 01 4C 00 08 6D 79 43 6F  Nameq.~..L..myCo
00000060   75 72 73 65 74 00 10 4C 63 6F 6D 70 6C 65 78 2F  urset..Lcomplex/
00000070   43 6F 75 72 73 65 3B 78 70 00 00 00 16 74 00 05  Course;xp....t..
00000080   53 61 73 68 61 74 00 06 50 65 74 72 6F 76 73 72  Sashat..Petrovsr
00000090   00 0E 63 6F 6D 70 6C 65 78 2E 43 6F 75 72 73 65  ..complex.Course
000000A0   00 00 00 00 00 00 00 01 02 00 02 4A 00 02 69 64  ...........J..id
000000B0   4C 00 04 6E 61 6D 65 71 00 7E 00 01 78 70 00 00  L..nameq.~..xp..
000000C0   00 00 00 00 0B 74 00 09 4A 61 76 61 20 42 61     .....t..Java Ba
000000D0   73 65                                            se
```

0x73: TC_OBJECT - pointer that the new object is serialized

0x72: TC_CLASSDESC - serialized object class pointer

00 0E: class filename length from application root - 14 characters

663 6F 6D 70 6C 65 78 2E 43 6F 75 72 73 65: complex.Course class

00 00 00 00 00 00 00 01: serialVersionUID = 1L

0x02: Flags value 02 means the object supports serialization

00 02: number of fields of this class

Serialization file format -2/4

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   AC ED 00 05 73 72 00 0F 63 6F 6D 70 6C 65 78 2E  ¬н..sr..complex.
00000010   53 74 75 64 65 6E 74 00 00 00 00 00 00 00 01 02  Student.........
00000020   00 04 49 00 03 61 67 65 4C 00 09 66 69 72 73 74  ..I..ageL..first
00000030   4E 61 6D 65 74 00 12 4C 6A 61 76 61 2F 6C 61 6E  Namet..Ljava/lan
00000040   67 2F 53 74 72 69 6E 67 3B 4C 00 08 6C 61 73 74  g/String;L..last
00000050   4E 61 6D 65 71 00 7E 00 01 4C 00 08 6D 79 43 6F  Nameq.~..L..myCo
00000060   75 72 73 65 74 00 10 4C 63 6F 6D 70 6C 65 78 2F  urset..Lcomplex/
00000070   43 6F 75 72 73 65 3B 78 70 00 00 00 16 74 00 05  Course;xp....t..
00000080   53 61 73 68 61 74 00 06 50 65 74 72 6F 76 73 72  Sashat..Petrovsr
00000090   00 0E 63 6F 6D 70 6C 65 78 2E 43 6F 75 72 73 65  ..complex.Course
000000A0   00 00 00 00 00 00 00 01 02 00 02 4A 00 02 69 64  ...........J..id
000000B0   4C 00 04 6E 61 6D 65 71 00 7E 00 01 78 70 00 00  L..nameq.~..xp..
000000C0   00 00 00 00 00 0B 74 00 09 4A 61 76 61 20 42 61  ......t..Java Ba
000000D0   73 65                                           se
```

Description of the first field of the class :

0x4a: variable type code - 'J', LONG ('B' for byte, 'C' for char, 'D' for double, 'F' for float, 'I' for int, 'L' for non-array object types, 'S' for short, 'Z' for boolean, and '[' for arrays))

00 02: variable name length

69 64: variable name - id

Serialization file format -3/4

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   AC ED 00 05 73 72 00 0F 63 6F 6D 70 6C 65 78 2E  ¬н..sr..complex.
00000010   53 74 75 64 65 6E 74 00 00 00 00 00 00 00 01 02  Student.........
00000020   00 04 49 00 03 61 67 65 4C 00 09 66 69 72 73 74  ..I..ageL..first
00000030   4E 61 6D 65 74 00 12 4C 6A 61 76 61 2F 6C 61 6E  Namet..Ljava/lan
00000040   67 2F 53 74 72 69 6E 67 3B 4C 00 08 6C 61 73 74  g/String;L..last
00000050   4E 61 6D 65 71 00 7E 00 01 4C 00 08 6D 79 43 6F  Nameq.~..L..myCo
00000060   75 72 73 65 74 00 10 4C 63 6F 6D 70 6C 65 78 2F  urset..Lcomplex/
00000070   43 6F 75 72 73 65 3B 78 70 00 00 00 16 74 00 05  Course;xp....t..
00000080   53 61 73 68 61 74 00 06 50 65 74 72 6F 76 73 72  Sashat..Petrovsr
00000090   00 0E 63 6F 6D 70 6C 65 78 2E 43 6F 75 72 73 65  ..complex.Course
000000A0   00 00 00 00 00 00 00 01 02 00 02 4A 00 02 69 64  ...........J..id
000000B0   4C 00 04 6E 61 6D 65 71 00 7E 00 01 78 70 00 00  L..nameq.~..xp..
000000C0   00 00 00 00 0B 74 00 09 4A 61 76 61 20 42 61     .....t..Java Ba
000000D0   73 65                                            se
```

Description of the second field of the class (the type of reference variables is described in the format "field descriptor" L ClassName ;):

0x4c: start of reference variable declaration "L",

00 04: variable name length

6E 61 6D 65: variable name - name

0x71: TC_REFERENCE reference to an object already written to the stream (Ljava/lang/String;)

00 7e 00 01 ???

Serialization file format -4/4

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   AC ED 00 05 73 72 00 0F 63 6F 6D 70 6C 65 78 2E  ¬н..sr..complex.
00000010   53 74 75 64 65 6E 74 00 00 00 00 00 00 00 01 02  Student.........
00000020   00 04 49 00 03 61 67 65 4C 00 09 66 69 72 73 74  ..I..ageL..first
00000030   4E 61 6D 65 74 00 12 4C 6A 61 76 61 2F 6C 61 6E  Namet..Ljava/lan
00000040   67 2F 53 74 72 69 6E 67 3B 4C 00 08 6C 61 73 74  g/String;L..last
00000050   4E 61 6D 65 71 00 7E 00 01 4C 00 08 6D 79 43 6F  Nameq.~..L..myCo
00000060   75 72 73 65 74 00 10 4C 63 6F 6D 70 6C 65 78 2F  urset..Lcomplex/
00000070   43 6F 75 72 73 65 3B 78 70 00 00 00 16 74 00 05  Course;xp....t..
00000080   53 61 73 68 61 74 00 06 50 65 74 72 6F 76 73 72  Sashat..Petrovsr
00000090   00 0E 63 6F 6D 70 6C 65 78 2E 43 6F 75 72 73 65  ..complex.Course
000000A0   00 00 00 00 00 00 00 01 02 00 02 4A 00 02 69 64  ...........J..id
000000B0   4C 00 04 6E 61 6D 65 71 00 7E 00 01 78 70 00 00  L..nameq.~..xp..
000000C0   00 00 00 00 00 0B 74 00 09 4A 61 76 61 20 42 61  ......t..Java Ba
000000D0   73 65                                           se
```

0x78: TC_ENDBLOCKDATA - end of description

0x70: TC_NULL - there is no parent class (otherwise it would be its description)00 00 00 00 00 00 00 0B - course id field value

0x74: TC_STRING - reference variable type pointer

00 09: the length of the value of the reference variable in the format

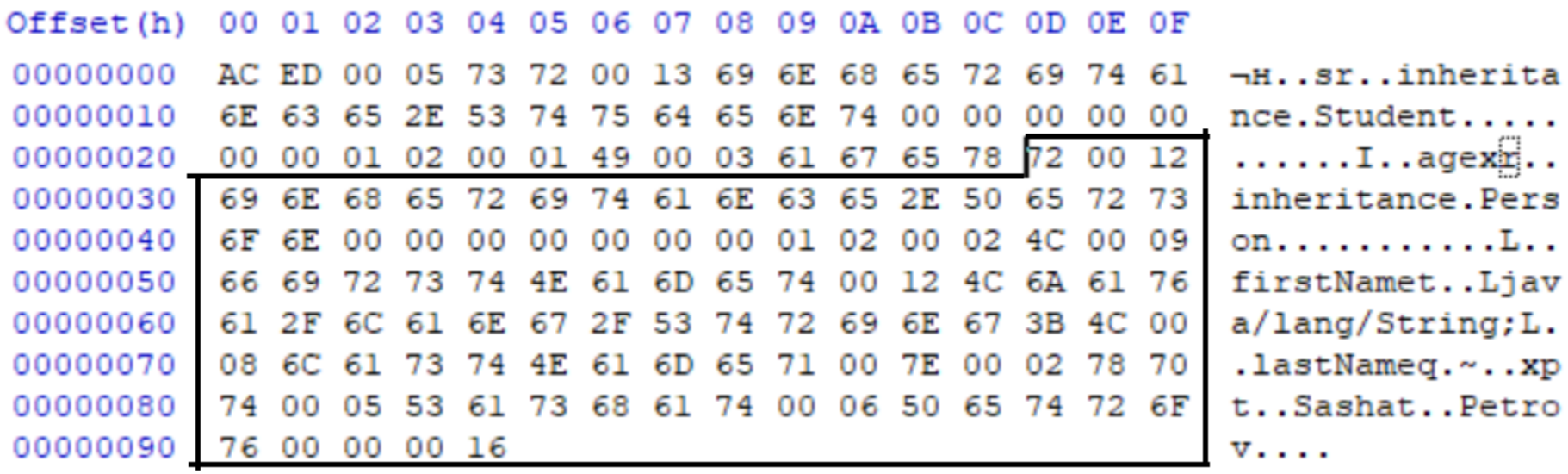
4A 61 76 61 20 42 61 73 65: reference variable value (Java Base)

Module contents

1. Java Serialization
   - Java - Serialization
   - Serializing an Object
   - The transient modifier
   - Complex Objects Serialization
   - Serialization with Inheritance
   - Custom Serialization in Java
   - Java Externalizable Interface

See serialization\inheritance

Serialization file format -1/4



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	AC	ED	00	05	73	72	00	13	69	6E	68	65	72	69	74	61	¬н..sr..inherita
00000010	6E	63	65	2E	53	74	75	64	65	6E	74	00	00	00	00	00	nce.Student.....
00000020	00	00	01	02	00	01	49	00	03	61	67	65	78	72	00	12	......I..agexr..
00000030	69	6E	68	65	72	69	74	61	6E	63	65	2E	50	65	72	73	inheritance.Pers
00000040	6F	6E	00	00	00	00	00	00	00	01	02	00	02	4C	00	09	on..........L..
00000050	66	69	72	73	74	4E	61	6D	65	74	00	12	4C	6A	61	76	firstNamet..Ljav
00000060	61	2F	6C	61	6E	67	2F	53	74	72	69	6E	67	3B	4C	00	a/lang/String;L.
00000070	08	6C	61	73	74	4E	61	6D	65	71	00	7E	00	02	78	70	.lastNameq.~..xp
00000080	74	00	05	53	61	73	68	61	74	00	06	50	65	74	72	6F	t..Sashat..Petro
00000090	76	00	00	00	16												v....

0x72: TC_CLASSDESC - serialized object class pointer

00 12: class filename length from application root - 18 characters

69 6E 68 65 72 69 74 61 6E 63 65 2E 50 65 72 73 6F 6E: inheritance.Person class

00 00 00 00 00 00 00 01: serialVersionUID = 1L

0x02: Flags value 02 means the object supports serialization

00 02: number of fields of this class

… description of the fields of the parent class and their values for the object

Module contents

1. Java Serialization
   - Java - Serialization
   - Serializing an Object
   - The transient modifier
   - Complex Objects Serialization
   - Serialization with Inheritance
   - Custom Serialization in Java
   - Java Externalizable Interface

Custom Serialization in Java 1/4

- Serializable - writeObject and readObject
- When we need to implement custom serialization we also have to define **in serialized class:**

```
private void writeObject(ObjectOutputStream out)
        throws IOException
```

Must be private!!!

and/or

```
private void readObject(ObjectInputStream in)
        throws IOException, ClassNotFoundException
```

and write and read all properties of the object.

- The JVM checks and calls these methods by the means of reflection

See serialization\custom

Module contents

1. Java Serialization
   - Java - Serialization
   - Serializing an Object
   - The transient modifier
   - Complex Objects Serialization
   - Serialization with Inheritance
   - Custom Serialization in Java
   - Java Externalizable Interface

Java Externalizable Interface

```
package java.io;

import java.io.ObjectOutput;
import java.io.ObjectInput;

public interface Externalizable extends java.io.Serializable {

   void writeExternal(ObjectOutput out) throws IOException;

   void readExternal(ObjectInput in) throws IOException,
                                            ClassNotFoundException;
}
```

This is not a marker interface

Java Externalizable Interface 1/5

- In contrast to Serializable interface, Externalizable delegates to the class the responsibility of how it should be serialized and deserialized.
- We are implementing he Externalizing interface by the class that should be serialized and override
  void writeExternal(ObjectOutput out) and
  void readExternal(ObjectInput in) methods
- We must also to define default constructor in the class (to pre-create an object during deserialization, from which the readExternal(ObjectInput in) method is then called).

See serialization\externalizable

Serializable vs Externalizable

- For tasks that do not require high performance or when serializing complex cross-referenced objects, it is best to use Serializable.
- On the other hand, when serialization speed is really important (for example, if you frequently (de)serialize a large number of objects), the Externalizable interface will benefit. Compare with basic deserialization.
- It is also worth noting that despite the speed advantage of the Externalizable mechanism (over standard serialization), when handling complex object structures, 'manual' serialization must be carefully thought out to avoid breaking the original object graph.

XML and JSON serialization

eXtensible Markup Language
Java Architecture for XML Binding
(JAXB, JSR-222)

Java API for JSON Processing
(JSON-P, JSR-353)

Alternatives to standard serialization

XML and JSON serialization

Third-party library:
FasterXML Jackson

File - Project Structure - Project Settings - Libraries +

```
<dependency>
   <groupId>com.fasterxml.jackson.dataformat</groupId>
   <artifactId>jackson-dataformat-xml</artifactId>
</dependency>

<dependency>
   <groupId>com.fasterxml.jackson.core</groupId>
   <artifactId>jackson-databind</artifactId>
</dependency>
```

See serialization\xmlserialization & serialization\jsonserialization