
DIASOFT
всё по-настоящему

DIASOFT
всё по-настоящему

# V. Ввод - вывод

## 5. Сериализация

# Сериализация объектов

Java предоставляет механизм называемый сериализацией объектов для представления объекта в виде последовательности байтов которая включает данные объекта, а также информацию о типе объекта и типах данных хранящихся в объекте. Сериализованный объект может быть записан в файл. После сериализации и записи объекта в файл он может быть прочитан из файла и десериализован. Вместе с объектом могут сериализоваться и десериализоваться все зависимые объекты. Для того чтобы объект класса можно было сериализовать и десериализовать необходимо чтобы класс реализовывал маркерный интерфейс Serializable. При десериализации объекта конструктор не вызывается.

Классы ObjectInputStream и ObjectOutputStream - высокоуровневые потоки которые содержат методы для сериализации и десериализации объектов.

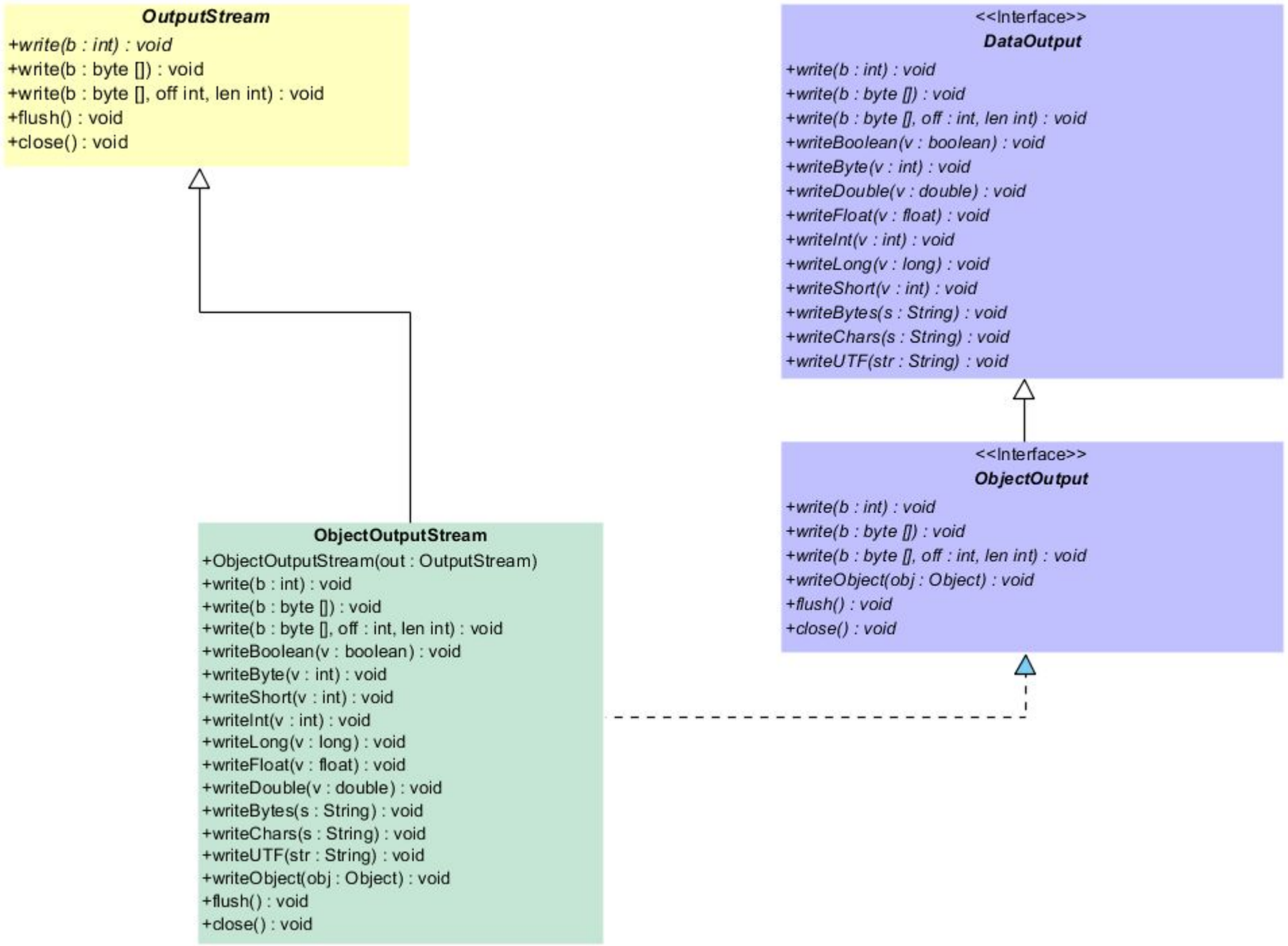
# Маркерный интерфейс Serializable

**Шаблон Хранитель:** Существует вариант реализации шаблона хранитель с помощью сериализации.

```
package java.io;

public interface Serializable {


}
```

Интерфейс Serializable должны реализовывать все классы предназначенные для сериализации. Это маркерный интерфейс - он не содержит ни одного метода.

# Объектный поток вывода

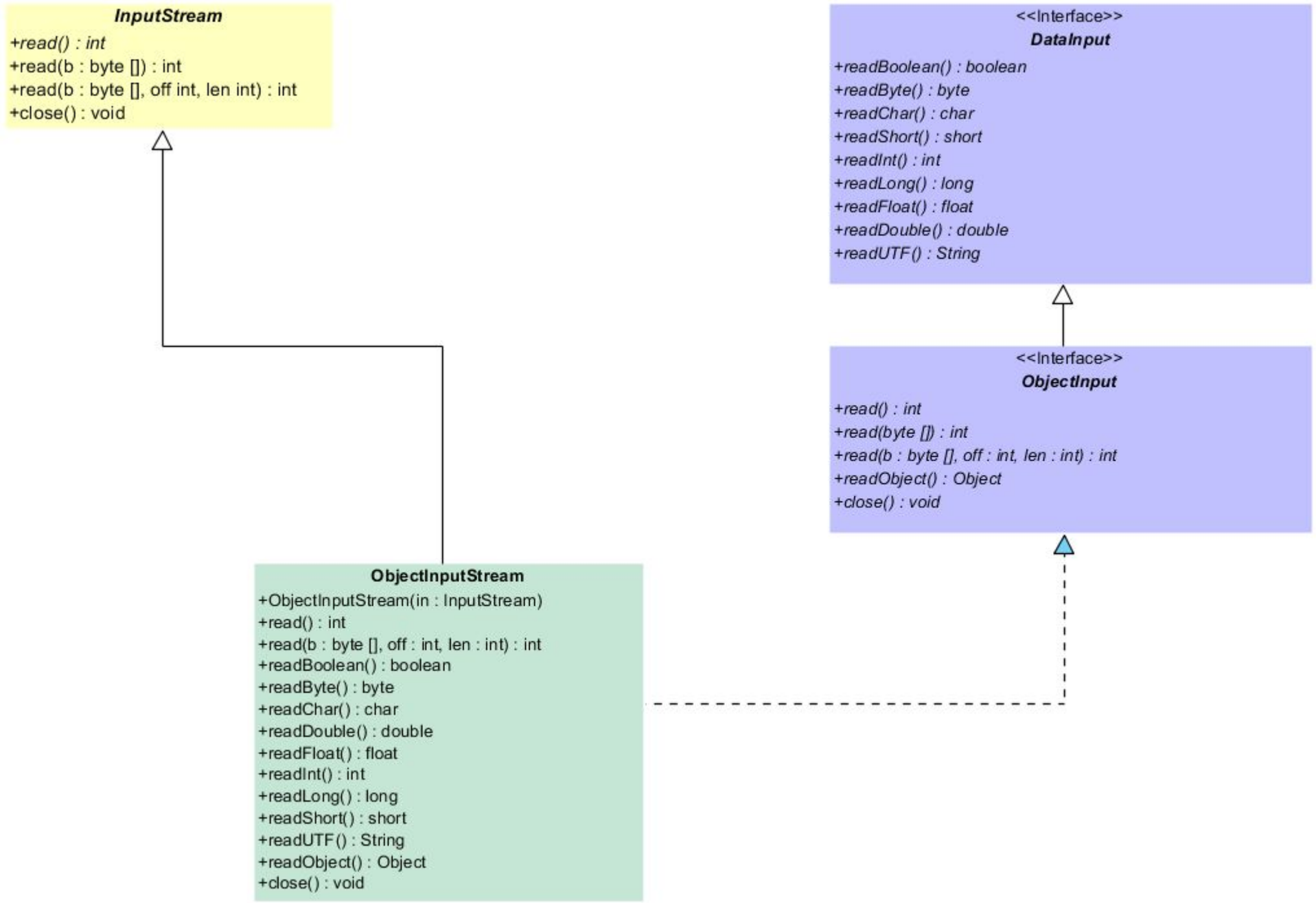


# Класс ObjectOutputStream

```
package java.io;

public class ObjectOutputStream extends OutputStream implements ObjectOutput, ObjectStreamConstants{

    public ObjectOutputStream(OutputStream out
    )
    public void writeObject(Object obj)

    public void write(int b)
    public void write(byte b[])
    public void write(byte b[], int off, int len)

    public final void writeBoolean(boolean v)
    public final void writeByte(int v)
    public final void writeShort(int v)
    public final void writeChar(int v)
    public final void writeInt(int v)
    public final void writeLong(long v)
    public final void writeFloat(float v)
    public final void writeDouble(double v)

    public final void writeBytes(String s)
    public final void writeChars(String s)
    public final void writeUTF(String str)

    public void flush()
    public void close()
}
```

| | |
|---|---|
| C | Класс ObjectOutputStream – высокоуровневый объектный поток содержащий метод для сериализации объектов. Он предназначен для превращения объектов в последовательность байтов и запись этой последовательности в байтовый поток вывода. Байтовый поток вывода задаётся в конструкторе. Класс также содержит методы для записи примитивов. |

# Объектный поток ввода



# Класс ObjectInputStream

```
package java.io;

public class ObjectInputStream extends InputStream implements ObjectInput, ObjectStreamConstants {

    public ObjectInputStream(InputStream in)

    public final Object readObject()

    public int read()
    public int read(byte b[], int off, int len)

    public final boolean readBoolean()
    public final byte readByte()
    public final short readShort()
    public final int readInt()
    public final long readLong()
    public final float readFloat()
    public final double readDouble()
    public final String readUTF()

}
```

Класс ObjectInputStream высокоуровневый объектный поток содержащий метод для десериализации объектов. Предназначен для превращения последовательности байтов считанной из байтового потока ввода в объект. Байтовый поток ввода задаётся в конструкторе.  Также содержит методы для чтения примитивов.

# Сериализация и десериализация одного объекта

# Сериализуемый класс

DIASOFT
всё по-настоящему

```
class Employee implements Serializable {

   public Employee() {
      System.out.println("Employee object is being created using a default constructor");
   }

   public Employee(String name, short yearOfBirth, char gender,
         boolean isMarried, int salary) {

      this.name = name;
      this.yearOfBirth = yearOfBirth;
      this.gender = gender;
      this.isMarried = isMarried;
      this.salary = salary;

      numEmployees++;
      System.out.println("Employee object is being created using a constructor");
   }

   public String toString() {
      return "Employee [name=" + name + ", yearOfBirth=" + yearOfBirth
            + ", gender=" + gender + ", isMarried=" + isMarried + ", salary="
            + salary + "]";
   }

   public static int getNumEmployees(){
      return numEmployees;
   }

   private String name;
   private short yearOfBirth;
   private char gender;
   private boolean isMarried;
   private int salary;

   private static int numEmployees = 0;
}
```

# Сериализация

DIASOFT
всё по-настоящему

```
public class ObjectOutputDemo {

   public static void main(String[] args) {

      Employee bob = new Employee("Robert", (short) 1987, 'M', true, 30000);
      System.out.println(bob);
      System.out.println("Total number of employees: " + Employee.getNumEmployees());

      FileOutputStream fos = null;
      ObjectOutputStream oos = null;

      try {
         fos = new FileOutputStream("I:\\FileIO\\employee.dat");
         oos = new ObjectOutputStream(fos);
         oos.writeObject(bob);
         System.out.println("Employee object was serialized");

      } catch (IOException e) {
         System.out.println("An I/O error occured");
      } finally {
         try {
            if (oos != null)
               oos.close();
         } catch (IOException e) {
            System.out.println("Error closing file");
         }
      }
   }
}
```

```
Employee object is being created using a constructor
Employee [name=Robert, yearOfBirth=1987, gender=M, isMarried=true, salary=30000]
Total number of employees: 1
Employee object was serialized
```

# Десериализация

DIASOFT
всё по-настоящему

```
class ObjectInputDemo {

   public static void main(String[] args) {

      FileInputStream fis = null;
      ObjectInputStream ois = null;

      try {

         fis = new FileInputStream("I:\\FileIO\\employee.dat");
         ois = new ObjectInputStream(fis);

         Employee emp = (Employee) ois.readObject();
         System.out.println(emp);
         System.out.println("Total number of employees: " + Employee.getNumEmployees());

      } catch (IOException e) {
         System.out.println("An I/O error occured");
      } catch (ClassNotFoundException e) {
         System.out.println("Class was not found");
      }
      finally {
         try {
            if (ois != null)
               ois.close();
         } catch (IOException e) {
            System.out.println("Error closing file");
         }
      }
   }
}
```

```
Employee [name=Robert, yearOfBirth=1987, gender=M, isMarried=true, salary=30000]
Total number of employees: 0
```

# Десериализация

При десериализации объекта конструктор класса не вызывается. Статические поля класса не сериализуются и не десериализуются.

# Сериализация графа объектов

# Сериализуемый класс со ссылкой

DIASOFT
всё по-настоящему

```
public class Person implements Serializable{

   public Person(String name) {
      this.name = name;
      System.out.println("Person constructor is called, name=" + name);
   }

   public Person() {
      this.name = "A person";
      System.out.println("Person parameterless constructor is called, name="
            + name);
   }

   public String getName() {
      return name;
   }

   public void setSpouse(Person value) {
      spouse = value;
   }

   public Person getSpouse() {
      return spouse;
   }

   public String toString() {
      return "[Person: name=" + name + " spouse=" + spouse.getName() + "]";
   }

   private String name;
   private Person spouse;
}
```

# Сериализация

DIASOFT
всё по-настоящему

```
public class TwoObjectsOutputDemo {

   public static void main(String[] args) {

      Person ann = new Person("Ann");
      Person bob = new Person("Bob");
      ann.setSpouse(bob);
      bob.setSpouse(ann);

      System.out.println(ann);
      System.out.println(bob);

      FileOutputStream fos = null;
      ObjectOutputStream oos = null;

      try {
         fos = new FileOutputStream("I:\\FileIO\\person.dat");
         oos = new ObjectOutputStream(fos);

         oos.writeObject(ann);

      } catch (IOException e) {
         System.out.println("An I/O error occured");
      } finally {
         try {
            if (oos != null)
               oos.close();
         } catch (IOException e) {
            System.out.println("Error closing file");
         }
      }
   }
}
```

# Сериализация

```
Person constructor is called, name=Ann
Person constructor is called, name=Bob
[Person: name=Ann spouse=Bob]
[Person: name=Bob spouse=Ann]
```

# Десериализация

DIASOFT
всё по-настоящему

```
class TwoObjectsInputDemo {

   public static void main(String[] args) {

      FileInputStream fis = null;
      ObjectInputStream ois = null;

      try {

         fis = new FileInputStream("I:\\FileIO\\person.dat");
         ois = new ObjectInputStream(fis);

         Person ann = (Person) ois.readObject();
         System.out.println(ann);
         System.out.println(ann.getSpouse());

      } catch (IOException e) {
         System.out.println("An I/O error occured");
      } catch (ClassNotFoundException e) {
         System.out.println("Class was not found");
      }
      finally {
         try {
            if (ois != null)
               ois.close();
         } catch (IOException e) {
            System.out.println("Error closing file");
         }
      }
   }
}
```

```
[Person: name=Ann spouse=Bob]
[Person: name=Bob spouse=Ann]
```

# Несериализуемые поля

# Несериализуемые поля

Все поля экземпляра класса независимо от модификатора доступа сериализуются. Но можно задать сериализацию и десериализацию только части полей класса. Для этого можно явно указать сериализуемые поля используя serialPersistentFields или использовать ключевое слово transient для обозначения несериализуемых полей.

# Класс с несериализуемым полем

DIASOFT
всё по-настоящему

```
class Salesman implements Serializable {

   public Salesman(String n, double s) {
      name = n;
      salary = s;
   }

   public String getName() {
      return name;
   }

   public double getSalary() {
      return salary;
   }

  public String toString() {
      return getClass().getSimpleName() + "[name=" + name + ",salary=" + salary
            + ",bonus=" + bonus + "]";
   }

   public void setBonus(double bonus) {
      this.bonus = bonus;
   }

   private String name;
   private double salary;

   private transient double bonus;
}
```

# Сериализация

DIASOFT
всё по-настоящему

```
public class TransientOutputDemo {

   public static void main(String[] args) {


      Salesman bob = new Salesman("Robert", 30000);
      bob.setBonus(10000);
      System.out.println(bob);

      FileOutputStream fos = null;
      ObjectOutputStream oos = null;

      try {
         fos = new FileOutputStream("I:\\FileIO\\salesman.dat");
         oos = new ObjectOutputStream(fos);

         oos.writeObject(bob);

      } catch (IOException e) {
         System.out.println("An I/O error occured");
      } finally {
         try {
            if (oos != null)
               oos.close();
         } catch (IOException e) {
            System.out.println("Error closing file");
         }
      }
   }
}
```

```
Salesman[name=Robert,salary=30000.0,bonus=10000.0 ]
```

# Десериализация

DIASOFT
всё по-настоящему

```
class TransientInputDemo {

   public static void main(String[] args) {

      FileInputStream fis = null;
      ObjectInputStream ois = null;

      try {

         fis = new FileInputStream("I:\\FileIO\\salesman.dat");
         ois = new ObjectInputStream(fis);

         Salesman bob = (Salesman) ois.readObject();
         System.out.println(bob);

      } catch (IOException e) {
         System.out.println("An I/O error occured");
      } catch (ClassNotFoundException e) {
         System.out.println("Class was not found");
      }
      finally {
         try {
            if (ois != null)
               ois.close();
         } catch (IOException e) {
            System.out.println("Error closing file");
         }
      }
   }
}
```

```
Salesman[name=Robert,salary=30000.0,bonus=0.0 ]
```

# UID

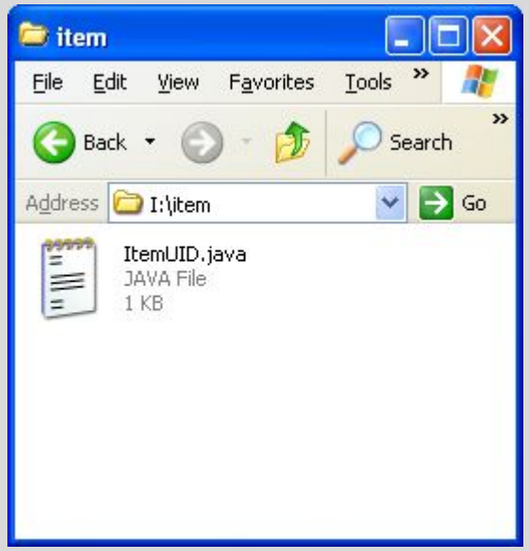
# UID

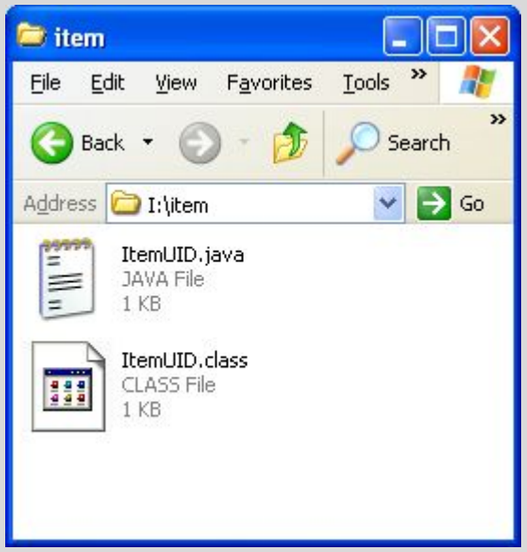


При сериализации объекта на основе информации о полях и методах класса создаётся идентификатор UID. Этот идентификатор записывается вместе с сериализованным объектом. При десериализации проверяется что UID сериализованного класса совпадает с UID класса. Если они различаются десериализация невозможна и выбрасывается исключение. UID могут различаться если после сериализации объекта класса в класс были внесены изменения.

# Получение UID

```
I:\item>dir
 Volume in drive I has no label.
 Volume Serial Number is 44AB-CB89

 Directory of I:\item

02/21/2013  12:14 PM    <DIR>          .
02/21/2013  12:14 PM    <DIR>          ..
02/21/2013  12:11 PM               476 ItemUID.java
               1 File(s)            476 bytes
               2 Dir(s)  48,619,651,072 bytes free

I:\item>javac ItemUID.java

I:\item>dir
 Volume in drive I has no label.
 Volume Serial Number is 44AB-CB89

 Directory of I:\item

02/21/2013  12:16 PM    <DIR>          .
02/21/2013  12:16 PM    <DIR>          ..
02/21/2013  12:16 PM               752 ItemUID.class
02/21/2013  12:11 PM               476 ItemUID.java
               2 File(s)          1,228 bytes
               2 Dir(s)  48,619,646,976 bytes free

I:\item>serialver ItemUID
ItemUID:    static final long serialVersionUID = -3358310746251373045L;

I:\item>
```



# serialVersionUID

Для того чтобы иметь возможность десериализовать сериализованный объект класса в объект изменённого класса можно использовать специальное поле serialVersionUID. Таким образом можно задать UID для класса и UID не будет меняться при внесении изменений в класс.

Eclipse может выдавать предупреждение или не компилировать код если класс реализует интерфейс Serializable, но не содержит поле serialVersionUID. Настроить эту опцию в Eclipse можно в Window > Preferences > Java > Compiler > Errors / Warnings > Potential Programming Problems. Если опция выдавать предупреждение установлена можно использовать предупреждения для автоматического добавления serialVersionUID.

# Класс с serialVersionUID

DIASOFT
всё по-настоящему

```
public class ItemUID implements Serializable{

   private static final long serialVersionUID = 52104546546023987 40L;

   public ItemUID(String name, int number) {
      this.name = name;
      this.number = number;
   }

   public String getName() {
      return name;
   }

   public int getNumber() {
      return number;
   }

   public String toString() {
      return "[name=" + name + ", number=" + number + "]";
   }

   private String name;
   private int number;
}
```

# Сериализация

DIASOFT
всё по-настоящему

```
public class UIDObjectOutputDemo {

   public static void main(String[] args) {

      ItemUID coffemaker = new ItemUID("Coffemaker", 2912);
      System.out.println(coffemaker);

      FileOutputStream fos = null;
      ObjectOutputStream oos = null;

      try {
         fos = new FileOutputStream("I:\\FileIO\\item.dat");
         oos = new ObjectOutputStream(fos);

         oos.writeObject(coffemaker);

      } catch (IOException e) {
         System.out.println("An I/O error occured");
      } finally {
         try {
            if (oos != null)
               oos.close();
         } catch (IOException e) {
            System.out.println("Error closing file");
         }
      }
   }
}
```

```
[name=Coffemaker, number=2912]
```

# Новый класс со старым serialVersionUID

DIASOFT
всё по-настоящему

```
public class ItemUID implements Serializable{

   private static final long serialVersionUID = 52104546546023987 40L;

   public ItemUID(String name, int number, String description) {
      this.name = name;
      this.number = number;
      this.description = description;
   }

   public String getName() {
      return name;
   }

   public int getNumber() {
      return number;
   }

   public String toString() {
      return "[name=" + name + ", number=" + number + ", description=" + description+"]";
   }

   private String name;
   private int number;
   private String description;
}
```

# Десериализация

DIASOFT
всё по-настоящему

```
class UIDObjectInputDemo {

   public static void main(String[] args) {

      FileInputStream fis = null;
      ObjectInputStream ois = null;

      try {

         fis = new FileInputStream("I:\\FileIO\\item.dat");
         ois = new ObjectInputStream(fis);

         ItemUID item = (ItemUID) ois.readObject();
         System.out.println(item);

      } catch (IOException e) {
         System.out.println("An I/O error occured");
      } catch (ClassNotFoundException e) {
         System.out.println("Class was not found");
      }
      finally {
         try {
            if (ois != null)
               ois.close();
         } catch (IOException e) {
            System.out.println("Error closing file");
         }
      }
   }
}
```

```
[name=Coffemaker, number=2912, description=null]
```

# readObject и writeObject

# readObject и writeObject

Для некоторых классов можно существенно улучшить сериализацию по умолчанию. Определив в классе методы readObject и writeObject можно изменить сериализацию объекта класса. Состояние сохраняется посредством записи полей по отдельности с помощью метода writeObject из класса ObjectOutputStream или методов для записи примитивных типов данных из DataOutput.

# Класс реализующий список с помощью массива

DIASOFT
всё по-настоящему

```
public class StringArrayList implements Serializable {

    private transient int size = 0;
    private transient String buf[] = new String[16];

    public void add(String s) {
        buf[size] = s;
        size++;
    }

    public String toString() {

        StringBuffer b = new StringBuffer();
        for (int i=0; i<size; i++) {
            b.append(buf[i]);
            b.append(" ");
        }
        return b.toString();
    }

    private void writeObject(ObjectOutputStream s) throws IOException {

        s.defaultWriteObject();
        s.writeInt(size);
        for (int i=0; i<size; i++)
            s.writeObject(buf[i]);
    }

    private void readObject(ObjectInputStream s) throws IOException, ClassNotFoundException {

        s.defaultReadObject();
        int size = s.readInt();
        for (int i = 0; i < size; i++)
            add((String)s.readObject());
    }
}
```

# Сериализация

DIASOFT
всё по-настоящему

```
public class CustomOutputDemo {

   public static void main(String[] args) {

      StringLinkedList sList = new StringLinkedList();
      sList.add("apple");
      sList.add("orange");
      sList.add("banana");
      System.out.println("List contents are: " + sList);

      FileOutputStream fos = null;
      ObjectOutputStream oos = null;

      try {
         fos = new FileOutputStream("I:\\FileIO\\fruitlist.dat");
         oos = new ObjectOutputStream(fos);

         System.out.println("Serializing .....");
         oos.writeObject(sList);

      } catch (IOException e) {
         System.out.println("An I/O error occured");
      } finally {
         try {
            if (oos != null)
               oos.close();
         } catch (IOException e) {
            System.out.println("Error closing file");
         }
      }
   }
}
```

```
List contents are: banana orange apple
Serializing .....
```

# Десериализация

DIASOFT
всё по-настоящему

```
class CustomInputDemo {

   public static void main(String[] args) {

      FileInputStream fis = null;
      ObjectInputStream ois = null;

      try {

         fis = new FileInputStream("I:\\FileIO\\fruitlist.dat");
         ois = new ObjectInputStream(fis);

         System.out.println("Deserializing .....");
         StringLinkedList sList = (StringLinkedList) ois.readObject();
         System.out.println("List contents are: " + sList);

      } catch (IOException e) {
         System.out.println("An I/O error occured");
      } catch (ClassNotFoundException e) {
         System.out.println("Class was not found");
      }
      finally {
         try {
            if (ois != null)
               ois.close();
         } catch (IOException e) {
            System.out.println("Error closing file");
         }
      }
   }
}
```

```
Deserializing .....
List contents are: apple orange banana
```

# Спасибо

**Россия, 127018,**
**Москва, ул. Полковая 3, стр. 14**
**Тел.: +7(495) 780 7575, 789 9339**
**Факс: +7(495) 780 7576, 789 9338**
**info@diasoft.ru, www.diasoft.ru**