Bài 17
File và Serialization

Module: ADVANCED PROGRAMMING WITH JAVA

Mục tiêu

- Trình bày được các lựa chọn khác nhau để thao tác với file nhị phân
- Thực hiện được các thao tác đọc và ghi file nhị phân
- Trình bày được khái niệm Serialization
- Triển khai được cơ chế Serialization trong java

Thao tác với File nhị phân

Dòng byte

Thao tác với file nhị phân

Dòng byte

- Dòng byte dành cho dữ liệu dạng nhị phân.
- Ví dụ:
  - Nếu 5 được lưu với dòng byte, nó sẽ được lưu trữ ở dạng nhị phân của số 5 là chuỗi bit 101.
- File được tạo bằng dòng byte là file nhị phân. File nhị phân được đọc bởi các chương trình biến đổi dữ liệu nhị phân ra định dạng con người đọc được.
- Các dòng dành cho việc xử lý dữ liệu nhị phân đều được kế thừa từ hai lớp trừu tượng InputStream và OutputStream.
- InputStream và OutputStream chỉ cung cấp các phương thức cho phép đọc/ghi dữ liệu thô ở dạng byte.
- Các lớp con của InputStream và OutputStream cho phép đọc/ghi các giá trị thuộc các kiểu dữ liệu phức tạp hơn.

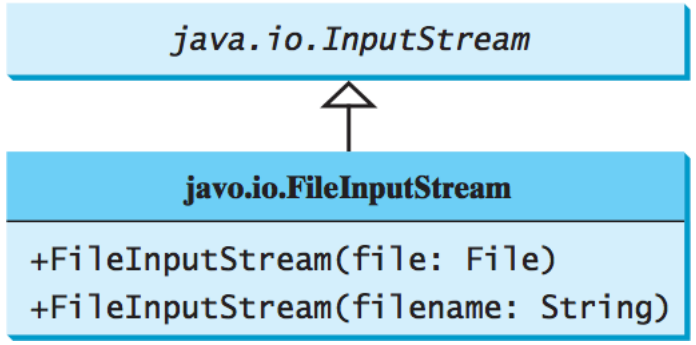
Thao tác với file nhị phân

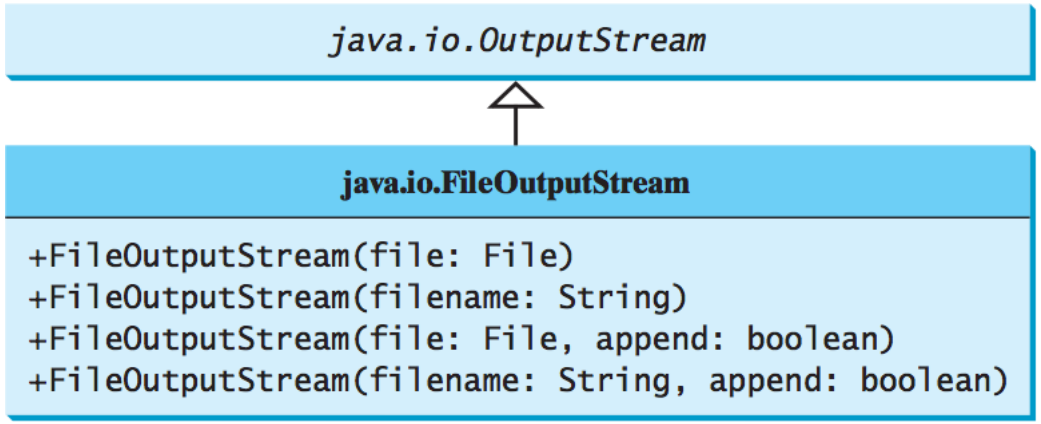
Một số dòng quan trọng để thao tác với file nhị phân

- FileInputStream/FileOutputStream
  - Dòng kết nối để nối trực tiếp với file nhị phân cần đọc/ghi theo dạng tuần tự.
- ObjectInputStream/ObjectOutputStream
  - Dòng nối tiếp, có thể nói với một InputStream/OutputStream khác, Các dòng này cho phép đọc/ghi từng đối tượng.

Lớp FileInputStream/FileOutputStream

- Lớp FileInputStream/FileOutputStream để thực hiện thao tác đọc/ghi dữ liệu từ file.





- Tất cả các phương thức trong các lớp thao tác với đọc/ghi trong gói java.io đều nén ra ngoại lệ java.io.Exception. Do đó cần khi sử dụng các phương thức này cần phải có cơ chếđể bắt hoặc ném lỗi.

```
public static void main(String[] args)
    throws IOException {
  // Perform I/O operations
}
```

```
public static void main(String[] args) {
  try {
    // Perform I/O operations
  }
  catch (IOException ex) {
    ex.printStackTrace();
  }
}
```

Ví dụ: Sử dụng FileOutputStream

```
public class TestDataOutputStream {
  public static void main(String args[]) {
    int a[] = {2, 3, 5, 7, 11};

    try {
      FileOutputStream fout = new FileOutputStream(args[0]);
      DataOutputStream dout = new DataOutputStream(fout);

      for (int i=0; i<a.length; i++)
        dout.writeInt(a[i]);
      dout.close();
    }
    catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

ghi dữ liệu kiểu int

Ví dụ: Sử dụng FileInputStream

```
public class TestDataInputStream {
  public static void main(String args[]) {
    try {
       FileInputStream fin = new FileInputStream(args[0]);
       DataInputStream din = new DataInputStream(fin);

       while (true) {
         System.out.println(din.readInt());
       }
    }
    catch (EOFException e) {
    }
    catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

đọc dữ liệu kiểu int

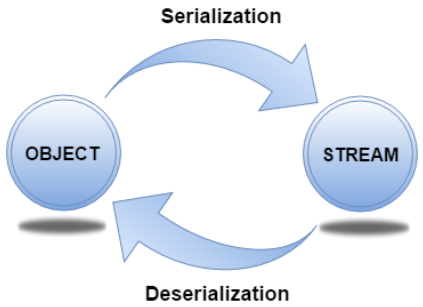
Serialization

Serialization là gì?

Serialization trong java

Serialization là gì?

- Serialization đơn giản chỉ là chuyển từ một object tồn tại thành một mảng byte.
- Mảng byte này đại diện cho class của object, phiên bản của object, và trạng thái của object.
- Mảng byte này có thể được sử dụng giữa các máy ảo JVM đang chạy cùng code truyền / đọc các object.
- Ngược lại Deserialization là quá trình xây dựng lại các byte vào một đối tượng.
- Java Serialization API cung cấp một cơ chế chuẩn cho các nhà phát triển để xử lý đối tượng.

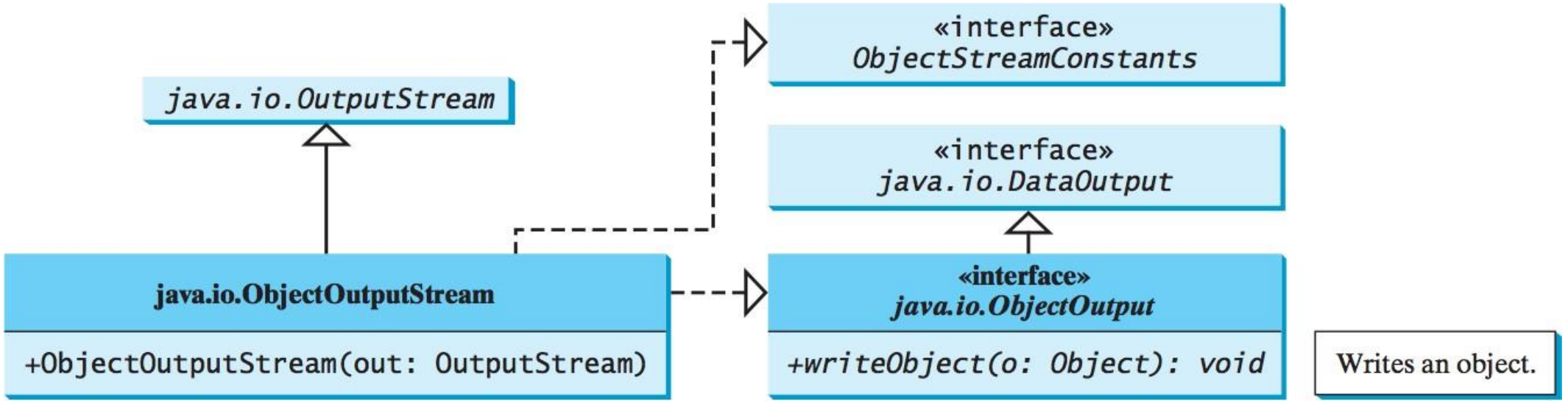


Serialization trong Java

- Trong java để mà serialize một object, bạn cần phải đảm bảo rằng class của object đó implements java.io.Serializable interface.
- Serializable interface không định nghĩa một phương thức nào, nó chỉ có nhiệm vụ chỉ ra đây là object có thể serialized được
- Hai lớp ObjectInputStream và ObjectOutputStream chứa các phương thức dùng để serializing và deserializing một object.
- ObjectOutputStream chứa phương thức writeObject(Object x) để serializes một đối tượng và gửi nó đến một output stream.
- ObjectInputStream chứa phương thức readObject() để lấy ra đối tượng từ stream và deserializes nó. Giá trị trả về là một object, vì vậy cần phải ép kiểu cho nó.

Lớp ObjectOutputStream

- Thao tác ghi đối tượng vào file



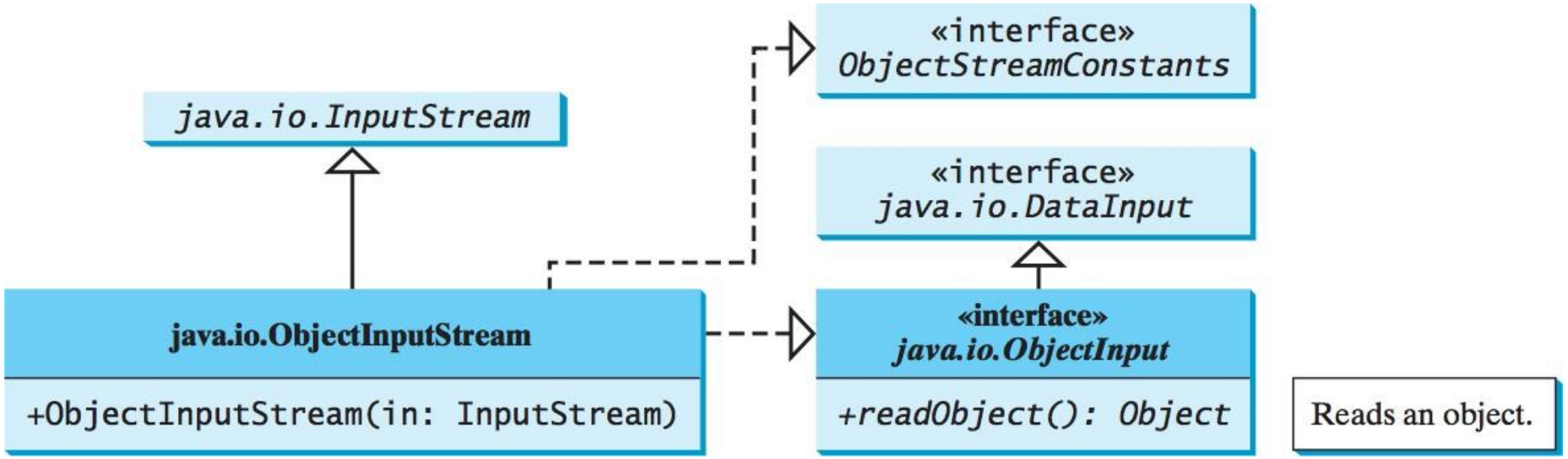
Ví dụ sử dụng lớp ObjectOutputStream

Ghi một chuỗi, số thực, đối tượng Date vào file object.dat
Lớp String/Double/Date đã được implement từ interface java.io.Serializable

```
public static void main(String[] args) throws IOException {
  try ( // Create an output stream for file object.dat
    ObjectOutputStream output =
      new ObjectOutputStream(new FileOutputStream("object.dat"));
  ) {
    // Write a string, double value, and object to the file
    output.writeUTF("John");
    output.writeDouble(85.5);
    output.writeObject(new java.util.Date());
  }
}
}
```

Lớp ObjectInputStream

- Thao tác đọc đối tượng từ file



Ví dụ sử dụng lớp ObjectInputStream

```
public static void main(String[] args)
  throws ClassNotFoundException, IOException {
  try ( // Create an input stream for file object.dat
    ObjectInputStream input =
      new ObjectInputStream(new FileInputStream("object.dat"));
  ) {
    // Read a string, double value, and object from the file
    String name = input.readUTF();
    double score = input.readDouble();
    java.util.Date date = (java.util.Date)(input.readObject());
    System.out.println(name + " " + score + " " + date);
  }
}
```

Tổng kết

- Các dòng dành cho việc xử lý dữ liệu nhị phân đều được kế thừa từ hai lớp trừu tượng InputStream và OutputStream.
- Lớp FileInputStream/FileOutputStream được kế thừa từ InputStream/OutputStream để thực hiện thao tác đọc/ghi dữ liệu từ file.
- Serialization đơn giản chỉ là chuyển từ một object tồn tại thành một mảng byte.
- Trong java để mà serialize một object, bạn cần phải đảm bảo rằng class của object đó implements java.io.Serializable interface.
- Hai lớp ObjectInputStream và ObjectOutputStream chứa các phương thức dùng để serializing và deserializing một object.

Hướng dẫn

- Hướng dẫn làm bài thực hành và bài tập
- Chuẩn bị bài tiếp: Threading