IT-Academy

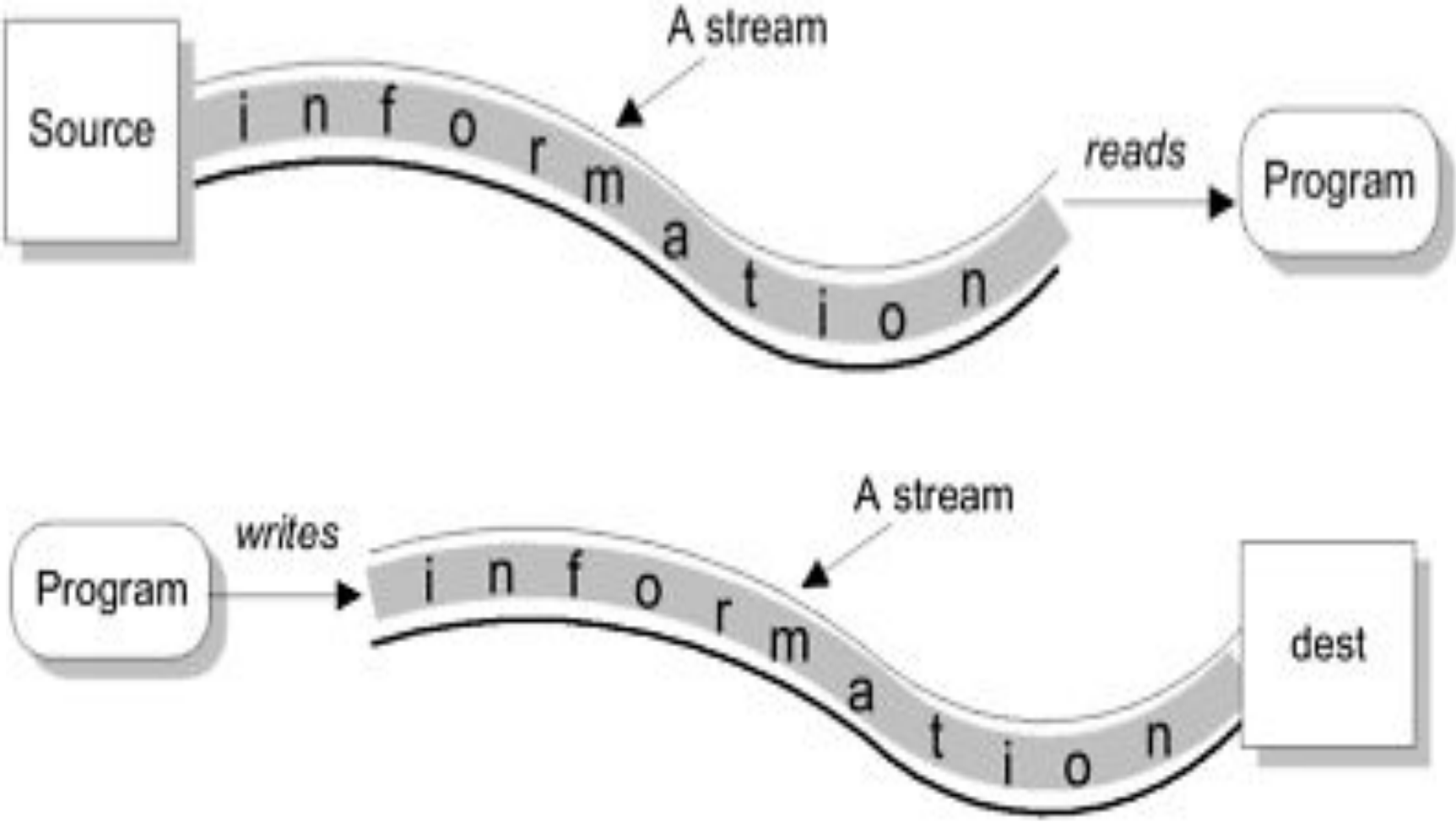
Основы ввода-вывода

Рассматриваемые вопросы

IT-Academy

- Reader и InputStream
- Writer и OutputStream
- Буферизация
- Класс Scanner
- Сериализация объектов

Общая концепция ввода-вывода



Reader и InputStream

IT-Academy

Reader и *InputStream* предоставляют схожий набор методов, но оперируют с различающимися типами данных.

InputStream содержит следующие методы для чтения байт и массивов байт:

- `int read();`
- `int read(byte cbuf[]);`
- `int read(byte cbuf[], int offset, int length);`

Reader определяет такие же методы, но для чтения символов и массивов символов:

- `int read();`
- `int read(char cbuf[]);`
- `int read(char cbuf[], int offset, int length);`

Writer и OutputStream

IT-Academy

Writer и OutputStream построены аналогично.

OutputStream определяет методы для записи байт и массивов байт:

- `int write(int c);`
- `int write(byte cbuf[]);`
- `int write(byte cbuf[], int offset, int length);`

Writer определяет методы для записи символов и массивов символов:

- `int write(int c);`
- `int write(char cbuf[]);`
- `int write(char cbuf[], int offset, int length);`

Все потомки – чтения и записи, символьный и байтовые – автоматически открываются при их создании. После использования необходимо принудительно закрывать потоки с помощью метода close().

Буферизация

IT-Academy

Классы для буферизация:

- BufferedReader
- BufferedWriter
- BufferedInputStream
- BufferedOutputStream

Буферизует данные при чтении или записи, тем самым уменьшая количество обращений к источнику данных. Буферизованные потоки как правило более эффективны, поэтому данные классы часто используют с другими потоками.

Буферизация

IT-Academy

Пример:

```
BufferedReader in = new BufferedReader(new FileReader("copyBytes.txt"));
BufferedWriter out = new BufferedWriter(new FileWriter("outCopyBytes.txt"));
```

Для чтения построчно используется метод *readLine().*

```
public String readLine();
```

Метод возвращает null при достижении конца файла.

Класс Scanner

IT-Academy

Для чтения из консоли используется класс *Scanner*.

Методы:

`boolean hasNext()` - для проверки наличия произвольной лексемы.

`boolean hasNext`*Тип*`()` - для проверки наличия конкретного типа.

Тип `next`*Тип*`()` – считывание данных конкретного типа.

`String next()` – считывание произвольной лексемы.

Serializable

IT-Academy

Сериализация - это процесс сохранения состояния объекта в последовательность байт;

Десериализация - это процесс восстановления объекта из этих байт.

Процесс сериализации заключается в сериализации каждого поля объекта, но только в том случае, если это поле не имеет спецификатора static или transient. Поля, помеченные ими не могут быть предметом сериализации.

Интерфейс Serializable

Для сериализации объекта класс должен реализовывать интерфейс *Serializable*.

```
public class Car implements Serializable {
 ......
}
```

Сериализация объекта

Для сериализации объекта используется класс ObjectOutputStream.

Конструктор `public ObjectOutputStream(OutputStream out)`

Пример:

```
FileOutputStream fos = new FileOutputStream("t.tmp");
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeInt(12345);
oos.writeObject("Today");
oos.writeObject(new Date());
oos.close();
```

Десериализация объекта

Для сериализации объекта используется класс ObjectInputStream.

Конструктор `public ObjectInputStream(InputStream in)`

Пример:

```
FileInputStream fis = new FileInputStream("t.tmp");
ObjectInputStream ois = new ObjectInputStream(fis);
int i = ois.readInt();
String today = (String) ois.readObject();
Date date = (Date) ois.readObject();
ois.close();
```

Вопросы

IT-Academy

IT-Academy

Спасибо за внимание