Serialization & File I/O

Pertemuan 11

Topics

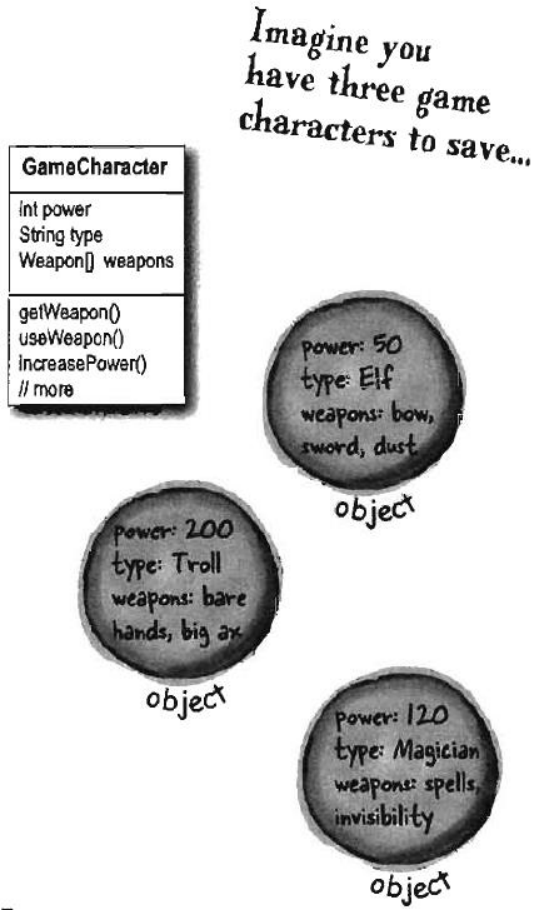
1. Serialization
2. File I/O

Pendahuluan

- Bayangkan anda sedang bermain game (fantasy / adventure) yang memiliki level yang banyak dan memiliki progress perkembangan karakternya (stronger, smarter, weapons, dll)
- Apabila datanya tidak disimpan, maka dalam bermain game, anda akan selalu memulai “from the scratch”

Contd..

- Terlebih lagi apabila anda memainkan banyak karakter



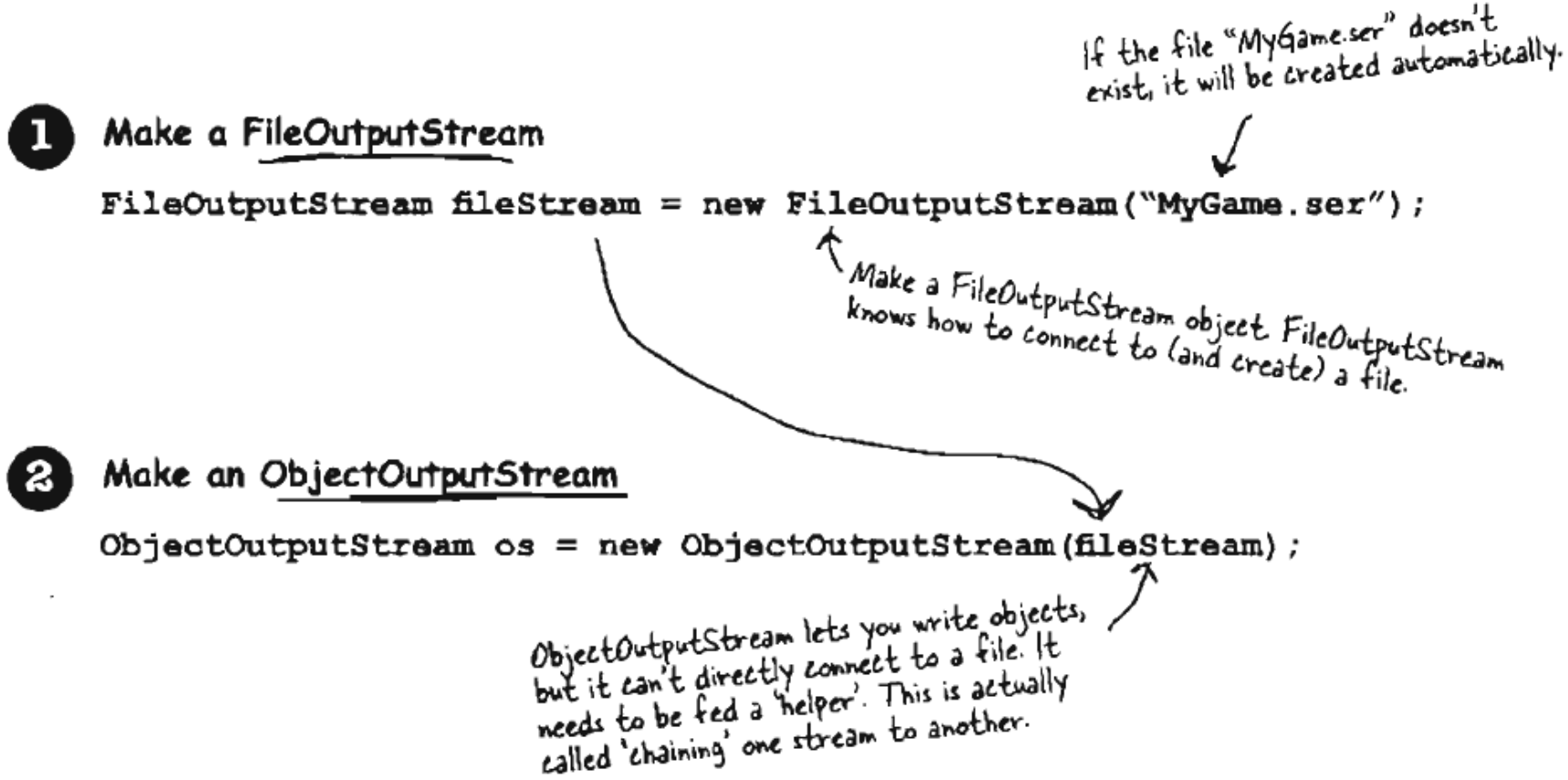
Saving object ?

- Penyimpanan suatu objek dapat dilakukan dengan 2(dua) cara :
  - Menggunakan serialization (hanya dapat dibaca oleh program java)
  - Menuliskan berupa plain text (dapat dibaca oleh program lainnya, seperti teks editor)

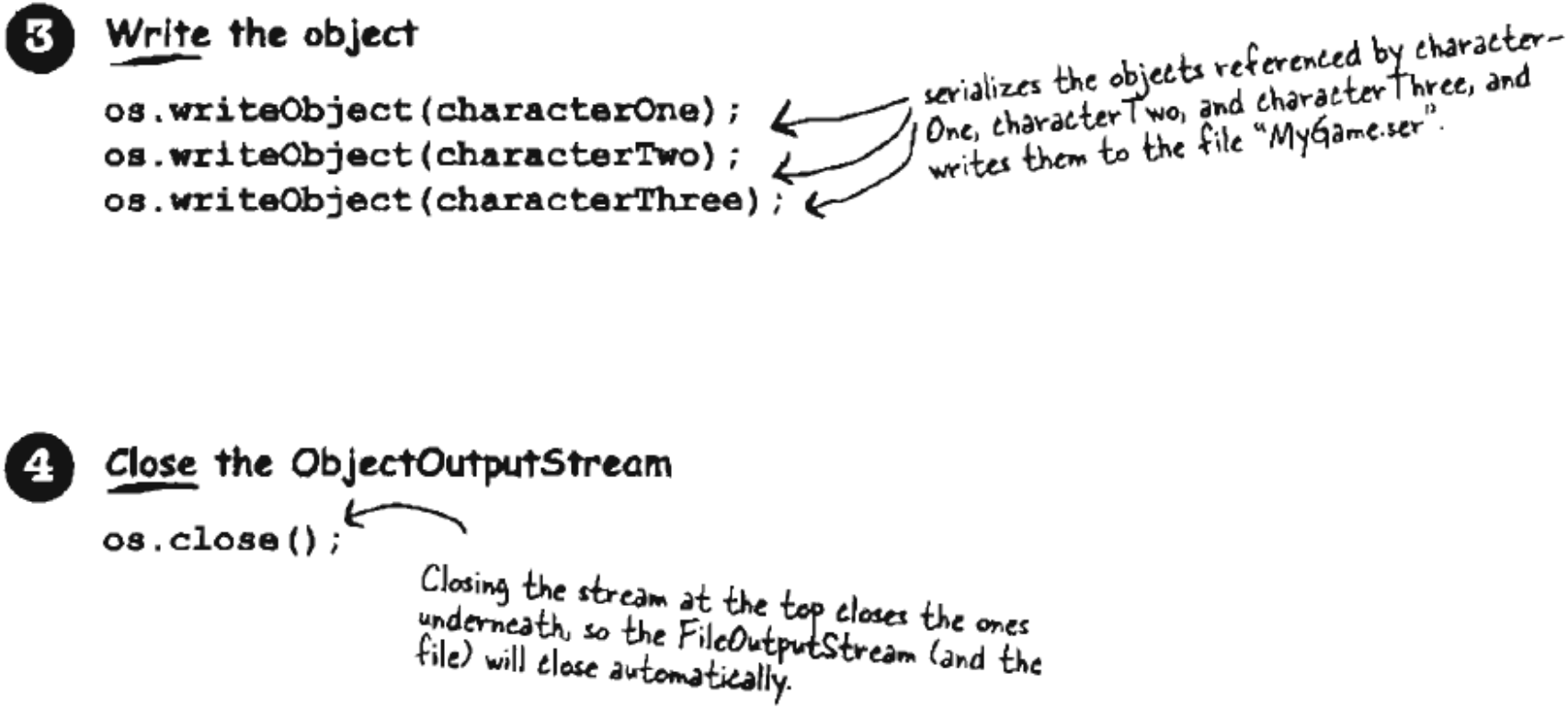
1. Serialization

Langkah Serialization

Serialization : saving object



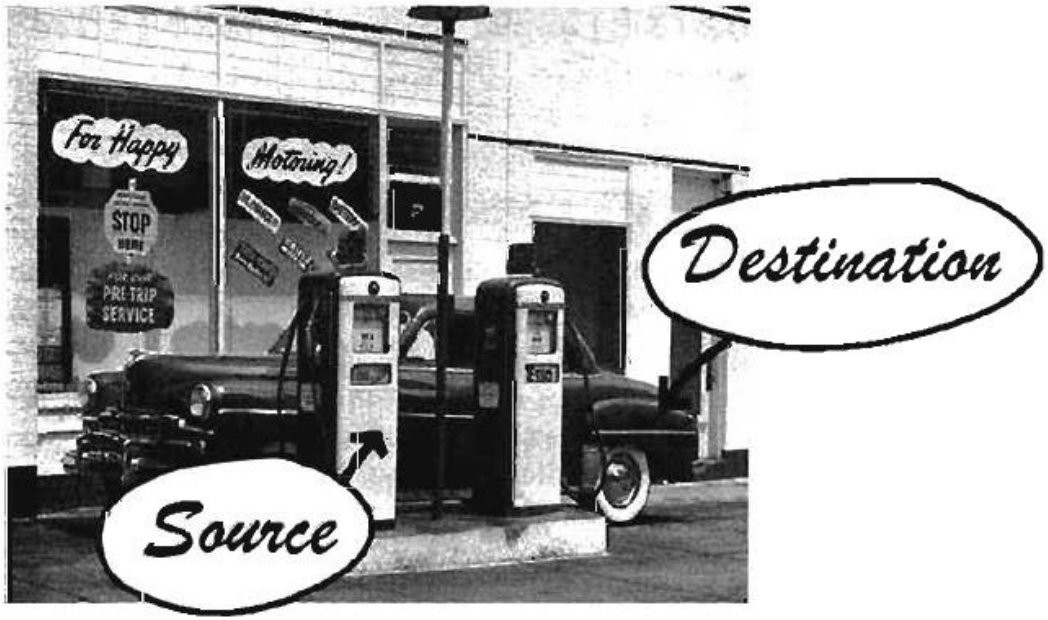
Langkah Serialization (contd..)



Stream

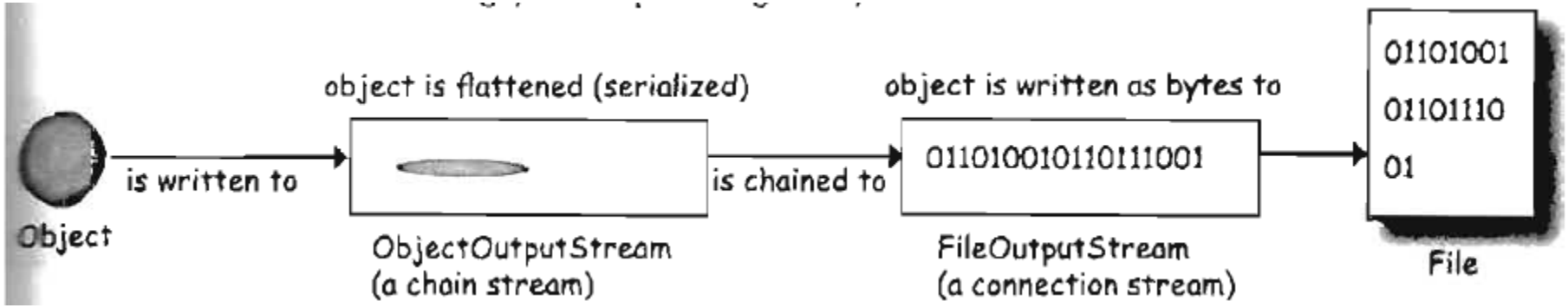
- Stream merepresentasikan suatu koneksi dari sumber (source) ke targetnya (destination)
- Contoh stream

: file, socket, dll.



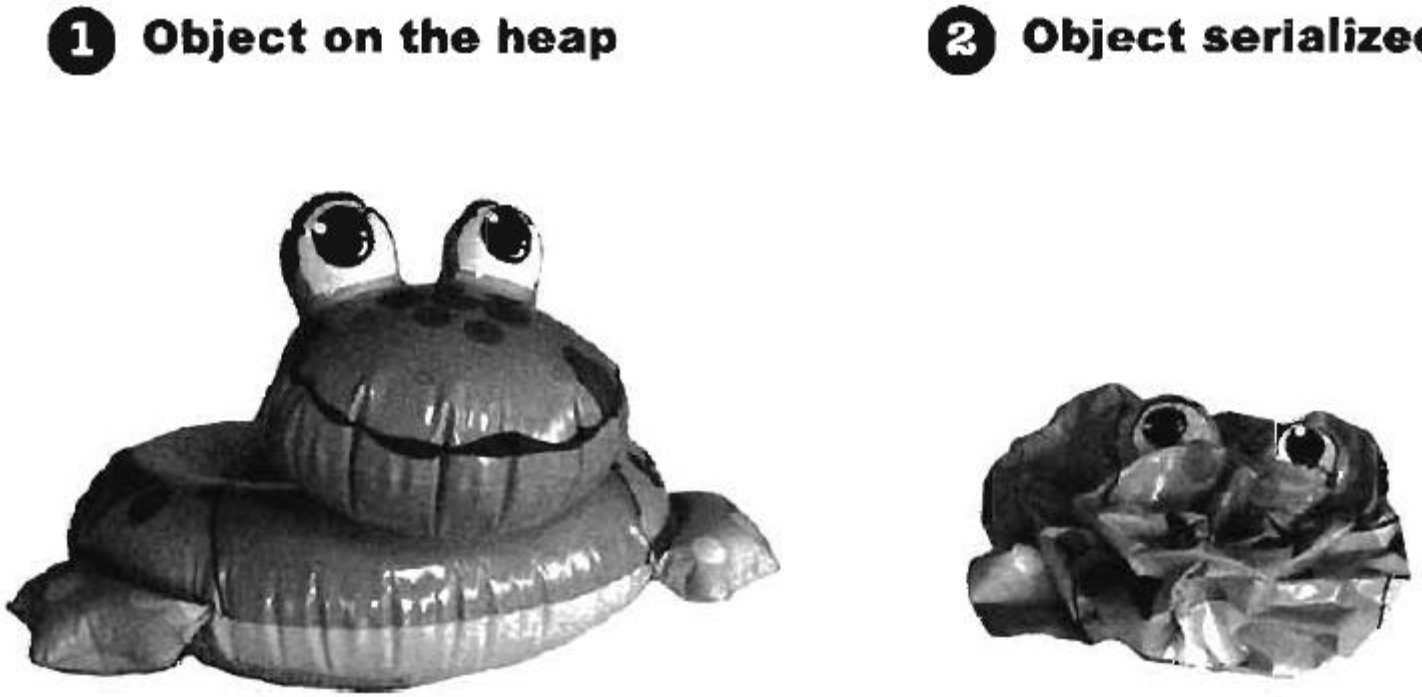
Stream (contd..)

- Apabila digambarkan secara sequence, maka alirannya adalah sbb :



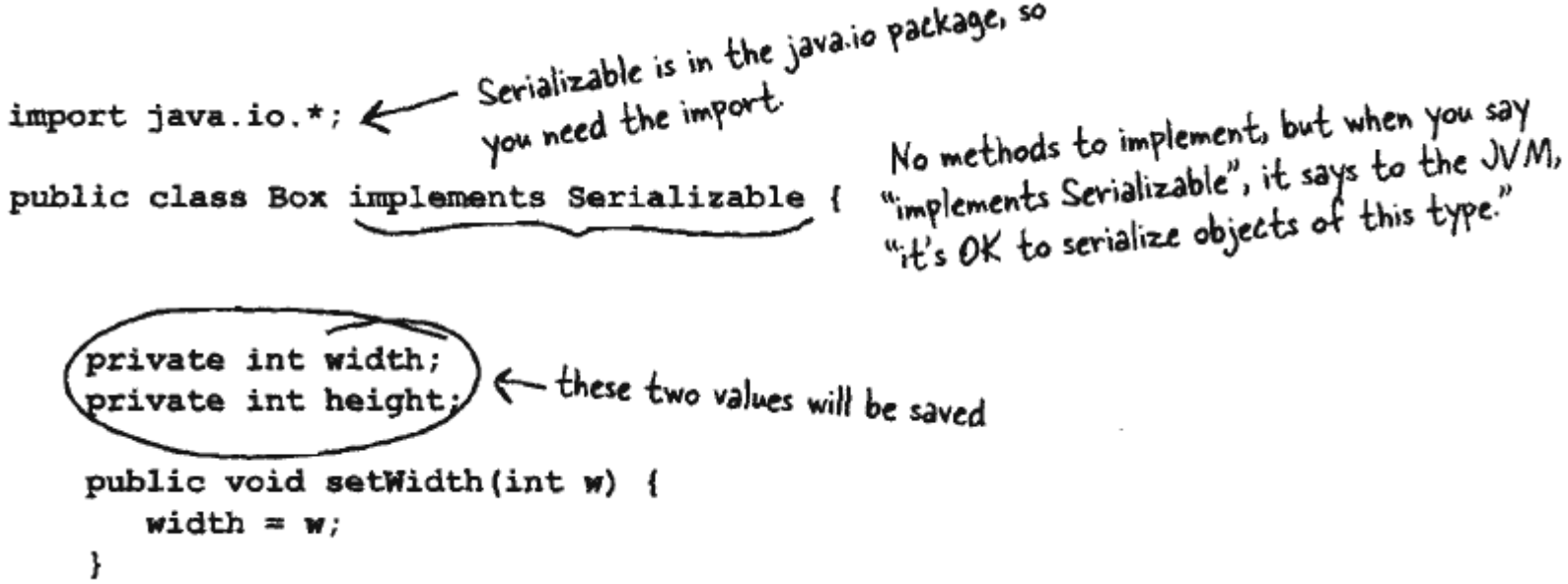
Object serialized ?

- Object is flattened



Cara class menjadi serialized ?

- Class yang dapat di-serialized, harus implement Serializable



Contd..

- Disimpan dalam ekstensi ".ser"

```
    public void setHeight(int h) {
       height = h;
    }

public static void main (String[] args) {

      Box myBox = new Box();
      myBox.setWidth(50);
      myBox.setHeight(20);

      try {
         FileOutputStream fs = new FileOutputStream("foo.ser");
         ObjectOutputStream os = new ObjectOutputStream(fs);
         os.writeObject(myBox);
         os.close();
      } catch(Exception ex) {
          ex.printStackTrace();
      }
   }
```

I/O operations can throw exceptions.

Connect to a file named "foo.ser" if it exists. If it doesn't, make a new file named "foo.ser".

Make an ObjectOutputStream chained to the connection stream.

Tell it to write the object.

Problem ? Serialization is all or nothing !

```
import java.io.*;

public class Pond implements Serializable {

    private Duck duck = new Duck();

    public static void main (String[] args) {
        Pond myPond = new Pond();
        try {
          FileOutputStream fs = new FileOutputStream("Pond.ser");
          ObjectOutputStream os = new ObjectOutputStream(fs);

          os.writeObject(myPond);
          os.close();

        } catch(Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

Pond objects can be serialized.

Class Pond has one instance variable, a Duck.

When you serialize myPond (a Pond object), its Duck instance variable automatically gets serialized.

its Duck instance variable refuses to be serialized (by not implementing Serializable).

When you try to run the main in class Pond:

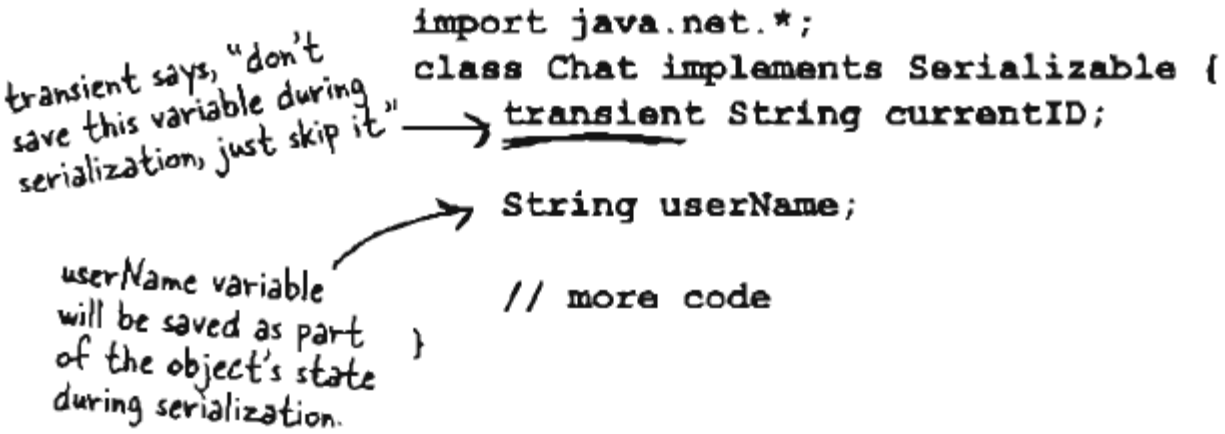
```
File Edit Window Help Regret
% java Pond
java.io.NotSerializableException: Duck
        at Pond.main(Pond.java:13)
```

```
public class Duck {
    // duck code here
}
```

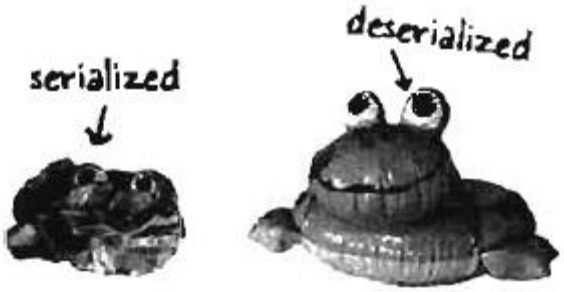
Yikes!! Duck is not serializable! It doesn't implement Serializable, so when you try to serialize a Pond object, it fails because the Pond's Duck instance variable can't be saved.

Transient in serialization

- Serialization mengharuskan semua objek didalamnya dapat di-serialize
- Apabila kita tidak ingin menyimpan objek kedalam serialization dapat menggunakan transient



Deserialization



- Deserialization : restoring object

1 Make a FileInputStream

```
FileInputStream fileStream = new FileInputStream("MyGame.ser");
```

If the file "MyGame.ser" doesn't exist, you'll get an exception.

Make a FileInputStream object. The FileInputStream knows how to connect to an existing file.

2 Make an ObjectInputStream

```
ObjectInputStream os = new ObjectInputStream(fileStream);
```

ObjectInputStream lets you read objects, but it can't directly connect to a file. It needs to be chained to a connection stream, in this case a FileInputStream.

Deserialization (contd..)

3 read the objects

```
Object one = os.readObject();
Object two = os.readObject();
Object three = os.readObject();
```

Each time you say readObject(), you get the next object in the stream. So you'll read them back in the same order in which they were written. You'll get a big fat exception if you try to read more objects than you wrote.

4 Cast the objects

```
GameCharacter elf = (GameCharacter) one;
GameCharacter troll = (GameCharacter) two;
GameCharacter magician = (GameCharacter) three;
```

The return value of readObject() is type Object (just like with ArrayList), so you have to cast it back to the type you know it really is.

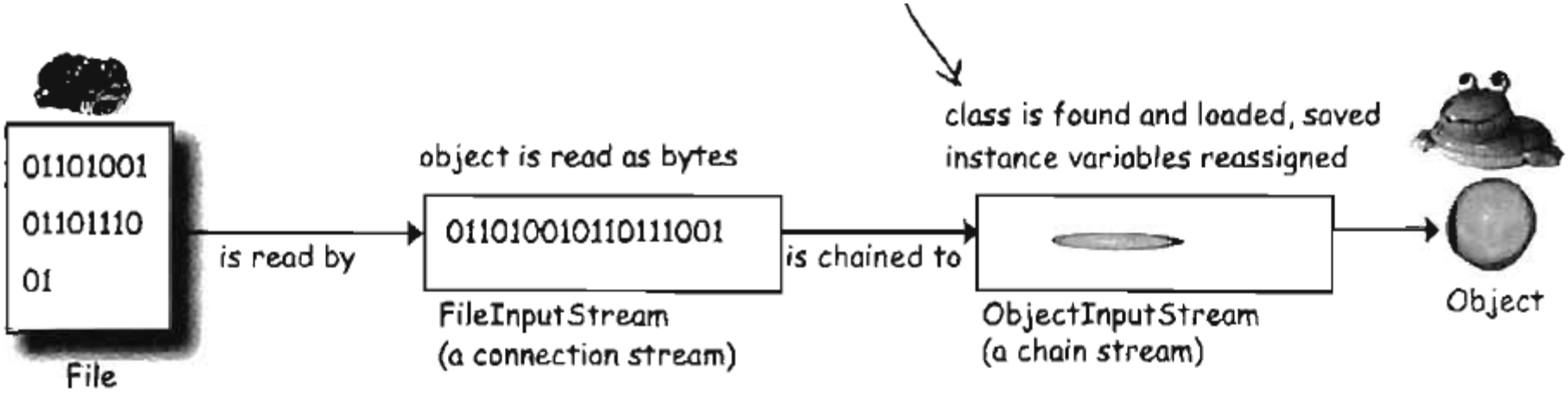
5 Close the ObjectInputStream

```
os.close();
```

Closing the stream at the top closes the ones underneath, so the FileInputStream (and the file) will close automatically.

Deserialization (contd..)

- Proses yang terjadi pada deserialization



2. Writing plain text

Writing plain text

- Serialization memungkinkan kita untuk menyimpan dan merestore data dengan mudah.
- Namun bagaimana kalau datanya ingin dibaca oleh aplikasi lain ? Simpan dalam bentuk plain text.

To write a serialized object:

```
objectOutputStream.writeObject(someObject);
```

To write a String:

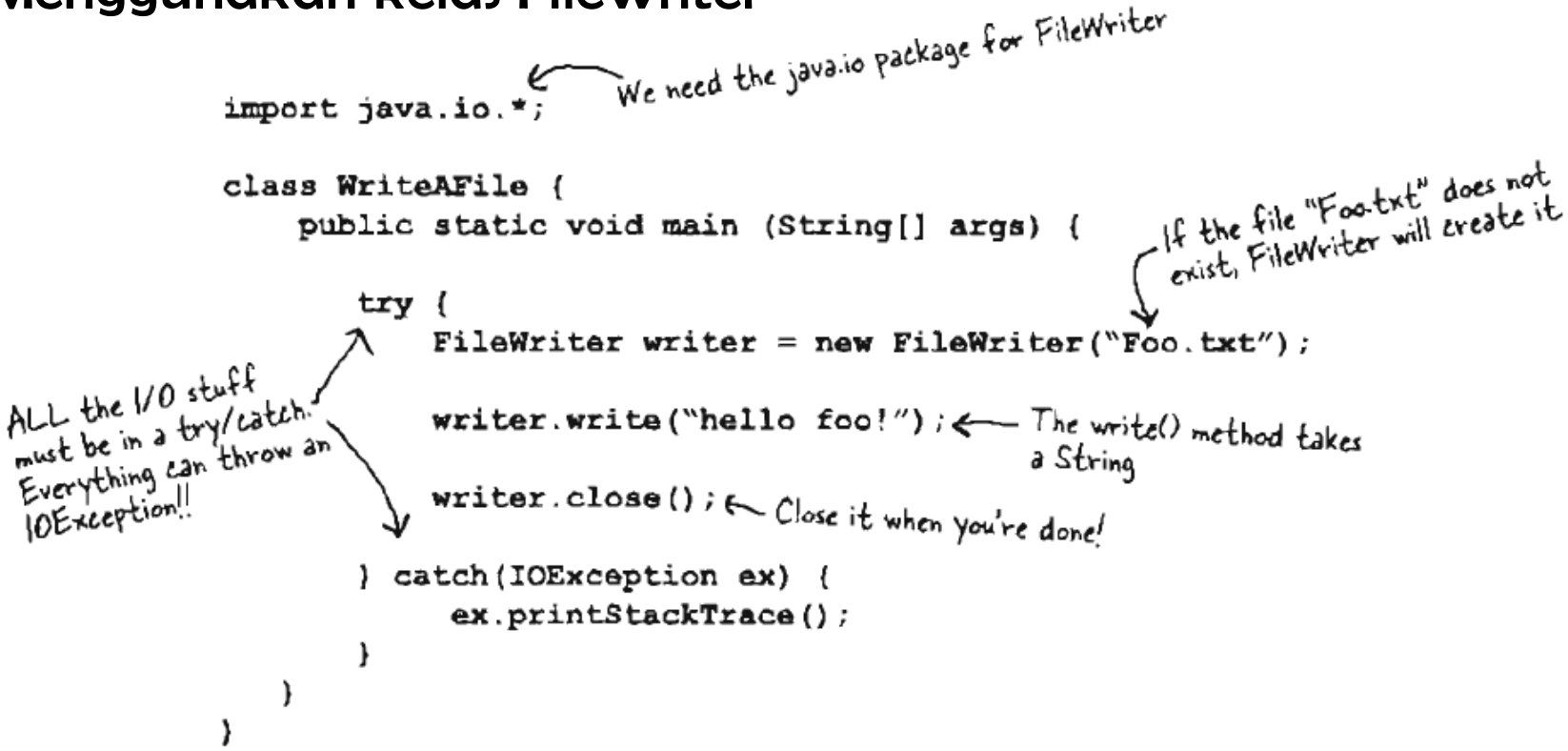
```
fileWriter.write("My first String to save");
```

What the game character data might look like if you wrote it out as a human-readable text file.

```
50,Elf,bow, sword,dust
200,Troll,bare hands,big ax
120,Magician,spells,invisibility
```


Contd..

- Data disimpan dalam ekstensi ".txt"
- Menggunakan kelas FileWriter



Reading plain text

- Untuk membaca plain text kita dapat menggunakan kelas FileReader
- Dengan parameter berupa lokasi file plain text-nya.
- Data dibaca baris perbaris melalui looping while (reader.readLine() != null) → akan terbaca terus menerus selama belum end of line

MyText.txt

```
import java.io.*;

class ReadAFile {
    public static void main (String[] args) {

        try {
            File myFile = new File("MyText.txt");
            FileReader fileReader = new FileReader(myFile);

            BufferedReader reader = new BufferedReader(fileReader);

            String line = null;

            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
            reader.close();

        } catch(Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

Don't forget the import

A FileReader is a connection stream for characters, that connects to a text file

Chain the FileReader to a BufferedReader for more efficient reading. It'll go back to the file to read only when the buffer is empty (because the program has read everything in it).

Make a String variable to hold each line as the line is read

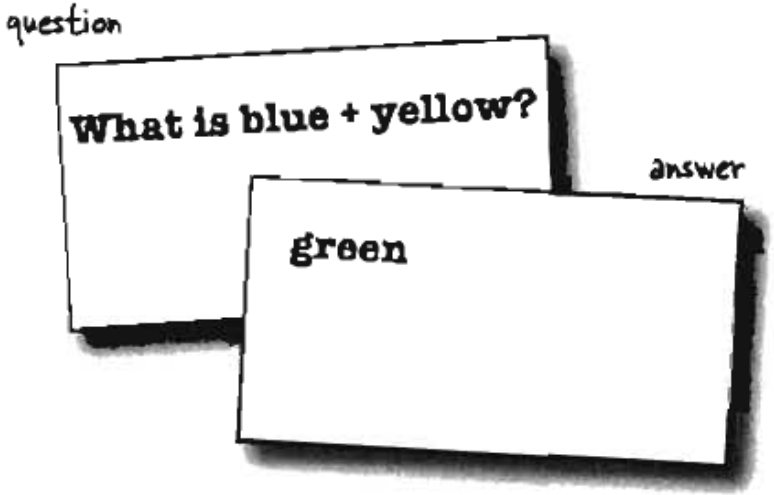
This says, "Read a line of text, and assign it to the String variable 'line'. While that variable is not null (because there WAS something to read) print out the line that was just read."

Or another way of saying it, "While there are still lines to read, read them and print them."

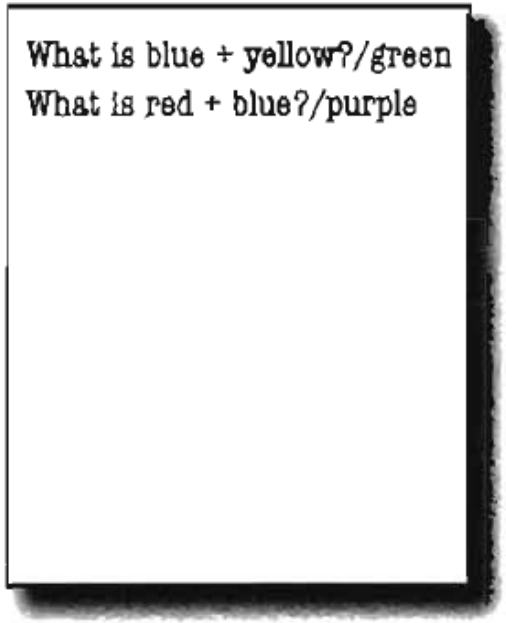
Parsing with string split()

- Data dapat dipecah menggunakan method split() berdasarkan separator-nya (/ , ; | dll)

Imagine you have a flashcard like this:



Saved in a question file like this:



Contd..

- String split let you break a string into pieces.


```
String toTest = "What is blue + yellow?/green";
String[] result = toTest.split("/");
for (String token:result) {
   System.out.println(token);
}
```

In the QuizCardPlayer app, this is what a single line looks like when it's read in from the file.

The split() method takes the "/" and uses it to break apart the String into (in this case) two pieces. (Note: split() is FAR more powerful than what we're using it for here. It can do extremely complex parsing with filters, wildcards, etc.)

Loop through the array and print each token (piece). In this example, there are only two tokens: "What is blue + yellow?" and "green".

Kesimpulan

- Saving object dapat dilakukan melalui 2(dua) cara, yaitu (1) using serialization, (2) saving into plain text.
- Serialization memungkinkan object disimpan dan direstore pada program java
- Plain text memungkinkan object dapat disimpan dalam plain text dan dibaca oleh teks editor lainnya.

Referensi

- Head first java 2nd Edition. Chapter 14 : Serialization & File I/O